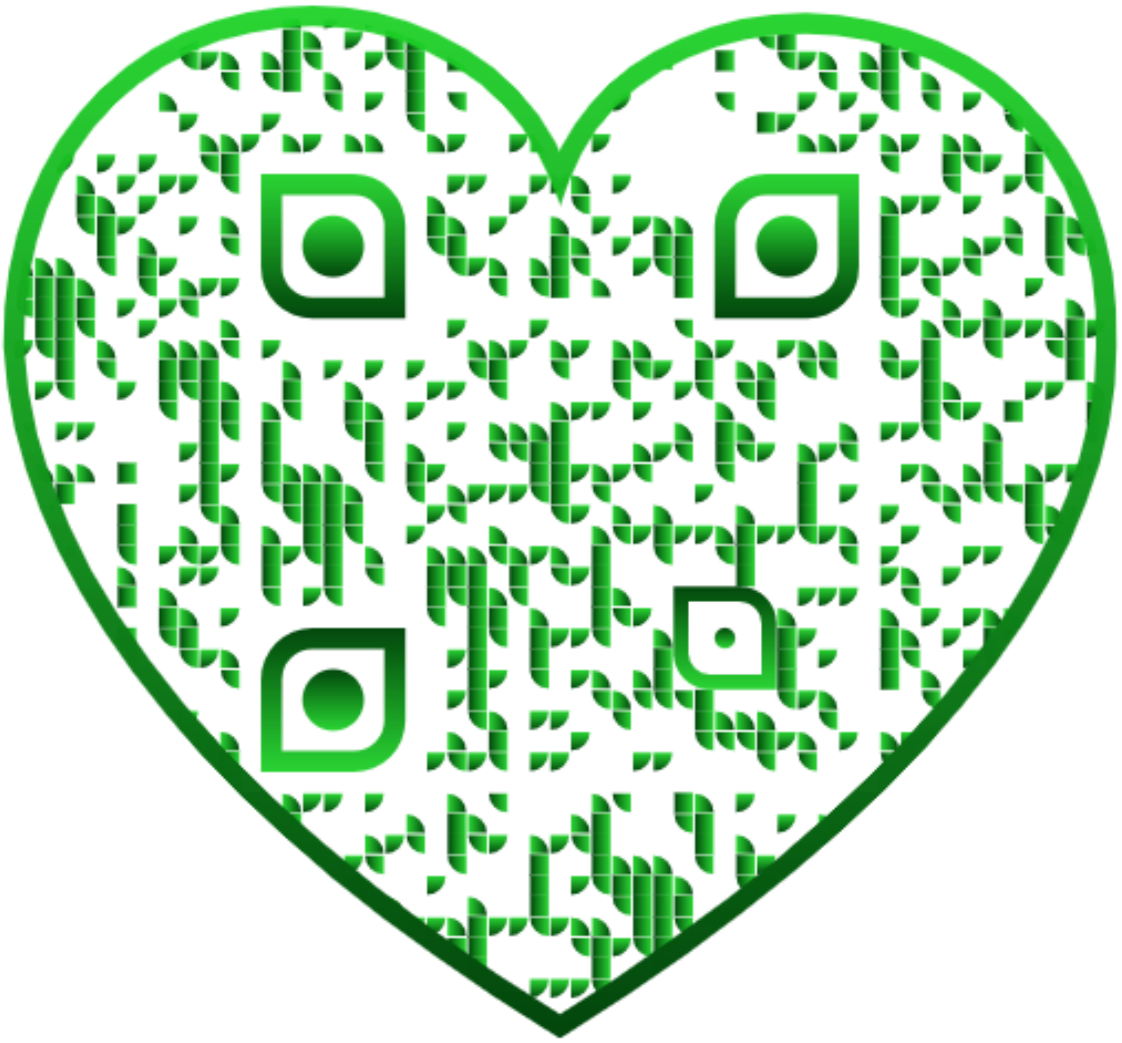
Applications mobiles avec Cordova et PhoneGap

Sébastien Pittion
Bastien Siebman

ght © 2015 Eyrolles.

EYROLLES



Applications mobiles avec Cordova et PhoneGap

Un seul développement pour un déploiement multiple

Maintenus par Adobe et la fondation Apache, Cordova et sa distribution PhoneGap sont des outils open source qui permettent de créer facilement et simultanément des applications mobiles pour plusieurs plates-formes du marché, comme iOS, Android, Windows Phone, et bien d'autres encore. En effet, il suffit aujourd'hui d'un seul développement pour assurer un déploiement multiple vers les différents magasins d'applications (App Store, Google Play Store, etc.). Utilisant les langages HTML, CSS et JavaScript, ces outils à la popularité croissante offrent une excellente alternative au code natif, notamment pour les développeurs web.

Concret et accessible, cet ouvrage est un recueil de bonnes pratiques et d'astuces pour réussir la création d'applications mobiles avec Cordova et PhoneGap, jusqu'à leur soumission sur les principaux magasins en ligne. Il comporte en outre une étude de cas complète, qui détaille la conception d'une application multi-plate-forme. Tout le code source des exemples est disponible en ligne sur https://github.com/siebmanb/rappelle-toi.

Au sommaire

Partie I : Premiers pas avec Cordova et PhoneGap. **Cordova ou PhoneGap ?** Un peu d'histoire • Cordova et PhoneGap en bref • **Prérequis et installation.** Phase « terminal » • Node.js • Kits de développement • Commande Line Interfaces • **Partie II : Développement d'une application. Création d'un projet.** Réfléchir avant de se lancer • Ne pas réinventer la roue • Architecture et structure • Versionner ses sources • **Conception et architecture d'une application.** Étapes de conception • Architecture de l'application • Bonnes pratiques et astuces • **Ajout de fonctionnalités spécifiques.** Installation et fonctionnement des plug-ins • Utilisation du répertoire merges • Utilisation des hooks • **Debug du code.** Dans un navigateur • Dans un simulateur • Sur un appareil • Accéder à un serveur local • Répercuter les changements sans compiler • **Partie III : Diffusion d'une application. Les magasins d'applications.** Publication sur l'App Store • Publication sur le Google Play Store • **PhoneGap Build.** Configuration via le fichier config.xml • Configuration via ConfiGAP • Utilisation.

À qui s'adresse ce livre ?

– À tous les acteurs d'un projet d'application mobile : décideurs, chefs de projet, développeurs, webdesigners...

– À ceux qui souhaitent concevoir, publier et vendre une application mobile pour iOS, Android, etc.

Sur le site https://github.com/siebmanb/rappelle-toi

– Téléchargez le code source de tous les exemples du livre

S. Pittion

Autodidacte évoluant dans le développement web depuis 15 ans, **Sébastien Pittion** est actuellement ingénieur développeur front-end chez Viseo. Suivant le projet Cordova/PhoneGap depuis le début, il a conçu et développé diverses applications mobiles avec ces outils, notamment dans le domaine de la télémédecine. Il est par ailleurs l'administrateur du compte Twitter @PhoneGapFR.
@fingerproof

B. Siebman

Ingénieur et entrepreneur, **Bastien Siebman** est le fondateur de l'agence ButterflyEffect, spécialisée dans le Web et le mobile solidaires, qui a récemment rejoint le groupe de communication CitizenRepublic. Il est également le cocréateur de l'outil Templana.com, bibliothèque de templates pour Asana.
@siebmanb

Copyright © 2015 Eyrolles.

Applications mobiles avec Cordova et PhoneGap

Copyright © 2015 Eyrolles.

Chez le même éditeur

F. Neuman. – **Le guide pratique iPhone et iOS8.**
N°14112, 2014, 160 pages.

Y. Garrett. – **Le guide pratique des tablettes Android.**
N°14070, 2014, 160 pages.

J.-M. Lacoste et T. Sarlandie. – **Programmation iOS 6 pour iPhone et iPad.**
N°13639, 2013, 352 pages.

J. Stark. – **Applications iPhone avec HTML, CSS et JavaScript.**
N°12745, 2010, 190 pages.

F. Daoust, D. Hazaël-Massieux. – **Relever le défi du Web mobile.** *Bonnes pratiques de conception et de développement.*
N°12828, 2011, 300 pages.

J. Chable, D. Guignard, E. Robles, N. Sorel. – **Programmation Android, 2e édition.**
N°13303, 2012, 520 pages.

E. Sarrion. – **jQuery mobile.** *La bibliothèque JavaScript pour le Web mobile.*
N°13388, 2012, 602 pages.

E. Marcotte. – **Responsive Web Design.**
N°13331, 2011, 160 pages.

L. Wroblewski. – **Mobile first.**
N°13406, 2012, 144 pages.

K. McGrane. – **Stratégie de contenu mobile.**
N°13675, 2013, 172 pages.

I. Canivet-Bourgaux. – **Référencement mobile.**
N°13667, 2013, 456 pages.

T. Baillet. – **Créer son premier thème WordPress pour mobile.**
N°13441, 2012, 128 pages.

Copyright © 2015 Eyrolles.

Applications mobiles avec Cordova et PhoneGap

Sébastien Pittion
Bastien Siebman

Copyright © 2015 Eyrolles.

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

Copyright © 2015 Eyrolles.

En application de la loi du 11 mars 1957, il est interdit de reproduire intégralement ou partiellement le présent ouvrage, sur quelque support que ce soit, sans l'autorisation de l'Éditeur ou du Centre Français d'exploitation du droit de copie, 20, rue des Grands Augustins, 75006 Paris.
© Groupe Eyrolles, 2015, ISBN : 978-2-212-14052-1

Préface

Depuis l'explosion du marché des smartphones et tablettes en 2007, les développeurs d'applications n'ont eu cesse de chercher des solutions techniques leur permettant de mutualiser leurs efforts de développement pour pouvoir adresser un maximum d'appareils du marché, tout en limitant la complexité inhérente au multi-plates-formes *(cross-platform)*. Grâce à la naissance du projet PhoneGap en 2009, puis au rachat de l'équipe technique par Adobe fin 2011, et enfin à la publication du cœur du projet sous licence libre avec pour nom de code Cordova, de nombreuses entreprises et indépendants ont misé sur cette technologie dite « hybride », en s'unissant pour faire de Cordova une plate-forme solide et pérenne pour leur stratégie mobile multi-plates-formes.

En choisissant les standards du Web pour créer vos applications et en vous basant sur Cordova pour les intégrer dans les écosystèmes natifs des différentes plates-formes, vous allez pouvoir former des équipes polyvalentes et performantes, capables d'intervenir aussi bien sur le développement web desktop que mobile. Vous pourrez ainsi capitaliser sur vos projets en mutualisant les compétences et le code source, et serez prêt par ailleurs pour adresser les futurs appareils (télévisions connectées, montres, lunettes…).

Pour ce faire, cet ouvrage va vous permettre d'aborder tous les aspects du développement d'applications mobiles multi-plates-formes avec Cordova. Vous y découvrirez que l'outil Cordova lui-même est relativement simple, l'essentiel du travail étant de produire des applications JavaScript en mode SPA (*Single Page Applications*, applications 100 % JavaScript, souvent créées avec des frameworks de type Backbone, AngularJS, Ionic, ReactJS…) bien intégrées, utiles pour vos utilisateurs et agréables à prendre en main.

Je vous souhaite une bonne découverte de la solution Cordova, et espère découvrir très rapidement vos applications sur les différents App Stores ☺.

Julien Bouquillon
Compte Twitter : @revolunet
Développeur, contributeur Cordova et organisateur de meetups

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Table des matières

AVANT-PROPOS 1

PREMIÈRE PARTIE

Premiers pas avec Cordova et PhoneGap 3

CHAPITRE 1

Cordova ou PhoneGap ? 5

Un peu d'histoire 5

HTML 5 5

L'App Store et consorts 6

HTML 5, le retour 7

Cordova en bref 7

Pourquoi ? 7

Comment ? 8

Pour qui ? 8

En pratique 9

PhoneGap en bref 11

Pourquoi ? 11

Quels avantages ? 12

En pratique 12

CHAPITRE 2

Prérequis et installation 15

Phase « terminal » 15

Shell 15

Bash 16

Émulateur de terminal 17

Entrer des commandes 19

Copyright © 2015 Eyrolles.

Aide et manuel d'utilisation 20
sudo 22
Node.js 23
Présentation 23
Installation de Node.js 24
Plusieurs installations en parallèle 25
Kits de développement 27
iOS 27
Android 30
Command Line Interfaces 37
Cordova CLI 38
PhoneGap CLI 39

DEUXIÈME PARTIE
Développement d'une application 41

CHAPITRE 3
Création d'un projet 43
Réfléchir avant de se lancer 43
Choisir les plates-formes cibles 43
Des étapes importantes avant le développement 44
Les plug-ins Cordova 46
Plug-in Cordova vs solution web 48
Ne pas réinventer la roue 50
Frameworks, bibliothèques et autres outils 50
Un choix important car structurant 52
Architecture et structure 53
Architecture d'un projet Cordova par défaut 53
Fichier de configuration 53
Application de démonstration 56
Utiliser un squelette personnalisé 57
Versionner ses sources 57
Pourquoi faire du versioning ? 57
Outils de versioning 58
Fonctionnement de Git 58
Git : commandes de base 59
Utiliser GitHub 60

Copyright © 2015 Eyrolles.

CHAPITRE 4

Conception et architecture d'une application 63
Étapes de conception 63
Étape 1 : définir le concept 63
Étape 2 : choisir les plates-formes cibles 64
Étape 3 : créer les wireframes 64
Étape 4 : identifier les fonctionnalités 65
Étape 5 : identifier les plug-ins 66
Étape 6 : créer le repository 66
Étape 7 : créer le projet Cordova 69
Architecture de l'application 70
config.xml, le fichier de configuration 70
Les fichiers CSS 71
Les polices d'écriture 71
Le fichier index.html 71
Le fichier main.js 74
Le fichier app.js 74
Ajouter une géolocalisation 75
Ajouter une photo 75
Ajouter une vidéo 76
Bonnes pratiques et astuces 77
Un projet bien planifié est à moitié fait 77
Tester, tester, tester 77
Être prêt à reconstruire le projet à tout moment 78
Chercher à mutualiser et éviter les processus manuels 78

CHAPITRE 5

Ajout de fonctionnalités spécifiques 79
Installer les plug-ins 79
Depuis un repository Cordova 79
Depuis un repository GitHub 79
Depuis un dossier en local 80
Avec plugman 80
Manuellement 80
Fonctionnement des plug-ins 80
Device 81

Copyright © 2015 Eyrolles.

SplashScreen 81
Camera 82
Geolocation 84
Dialogs 86
Network information 88
Battery Status 90
Status Bar 90
InAppBrowser 91
Utilisation du répertoire merges 94
Utilisation des hooks 95
Exemples de hooks 97

CHAPITRE 6
Debug du code 105
Dans un navigateur 105
Utiliser Chrome Dev tools 107
Émuler un appareil mobile 111
Dans un simulateur 113
Sur un appareil 123
Accéder à un serveur local 124
Répercuter les changements sans recompiler 125

TROISIÈME PARTIE
Diffusion d'une application 127

CHAPITRE 7
Les magasins d'applications 129
Publication sur l'App Store 129
Gestion des certificats sur Member Center 130
Publication via iTunes Connect 133
Déploiement ad hoc 139
Publication sur le Google Play Store 140
Création de l'APK 140
Création et mise en ligne de l'application 143

Copyright © 2015 Eyrolles.

CHAPITRE 8

PhoneGap Build 149
Présentation 149
Configuration via le fichier config.xml 151
Paramètres généraux 151
Personnalisation avancée 154
Icônes et splashscreens 156
Schémas d'URL 158
Fonctionnalités 158
Plug-ins 159
Sécurité 161
Configuration via ConfiGAP 162
Paramètres généraux 162
Paramètres avancés 163
Icônes et splashscreens 164
Permissions et informations 165
Vue Plugins 166
Utiliser PhoneGap Build 167
Via l'interface web 168
Debugging et Hydration 176
Via l'API REST 178
Via le terminal 178
Support et distribution 180

CONCLUSION 181

INDEX 183

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Avant-propos

Coder, c'est d'abord savoir parler un langage qui possède des règles propres, à l'instar de tous les langages. Qu'il s'agisse de mots, de ponctuation ou bien d'espacements, tous ces éléments syntaxiques s'avèrent incontournables pour tout développeur qui se respecte.

Car coder, c'est construire quelque chose, un programme destiné à une machine. Et pour ce faire, chaque développeur devra apprendre à maîtriser les outils dont il dispose, ainsi que, bien souvent, en découvrir de nouveaux. Compléter et dompter son vocabulaire sont donc deux des devoirs les plus importants d'un développeur. Après quoi, ce dernier pourra utiliser au mieux les outils mis à sa disposition. Il produira alors immanquablement de magnifiques poèmes emplis de figures stylistiques.

S'il est déjà relativement difficile de maîtriser un langage en particulier, l'univers mobile tel qu'on le connaît aujourd'hui fait pourtant appel à bon nombre d'entre eux. La fragmentation du marché (constructeurs, modèles, systèmes d'exploitation et bien évidemment langages de programmation différents) en est la cause directe. D'autant plus qu'il est en perpétuel changement. Alors, comment trouver son chemin, notamment quand on n'est encore qu'un développeur novice ? Pas de panique, vous découvrirez grâce à cet ouvrage qu'il est tout à fait possible de créer des applications mobiles destinées aux principaux acteurs du marché (Apple et Google) sans forcément se disperser. Et ce, grâce à ce formidable outil qu'est Cordova/PhoneGap.

Les plates-formes mobiles telles qu'Android ou iOS sont en pleine expansion depuis plusieurs années déjà. À mesure de leur évolution, notamment en termes de performances, le spectre des applications pouvant être développées avec Cordova/PhoneGap s'est considérablement élargi. Là où le développement d'un « simple » Tetris relevait auparavant du défi, mettre au point un jeu en 3D est maintenant à portée de main, avec de nombreux outils pour vous y aider. Certes, la technologie Cordova/PhoneGap n'est pas réellement destinée à produire des graphismes performants en trois dimensions, mais elle offre tout de même déjà beaucoup de possibilités.

Que vous souhaitiez réaliser un utilitaire, une version mobile d'un service web existant, ou même un jeu en 2D, Cordova/PhoneGap vous donnera rapidement accès à un résultat plus que satisfaisant et vous permettra de mettre celui-ci à disposition du plus grand nombre. Car c'est là la force de cet outil : développer pour plusieurs plates-formes dont la liste ne cesse de s'allonger : iOS, Android, Windows Phone, Firefox OS…

Copyright © 2015 Eyrolles.

Note

On parle dans ce cas de développement *cross-platform,* ce terme désignant le fait de ne maintenir qu'une seule base de code exécutable sur différents systèmes d'exploitation et machines. Vous verrez bien entendu dans cet ouvrage comment cela est possible.

Nous avons organisé ce livre en trois parties : histoire, présentation et installation de Cordova/PhoneGap, développement de votre première application, et mise à disposition auprès du grand public. Pour illustrer au mieux chaque étape de ce processus, nous avons conçu et développé, spécifiquement pour ce livre, une application volontairement simple, dont le code source, ouvert à tous, est hébergé sur GitHub (https://github.com/siebmanb/rappelle-toi). Vous pourrez ainsi vous y référer au cours de la lecture de cet ouvrage ou bien même participer à son amélioration !

Copyright © 2015 Eyrolles.

Partie I

Premiers pas avec Cordova et PhoneGap

Un nouveau monde s'ouvre à vous ! En effet, vous avez décidé de vous lancer dans le développement d'applications mobiles et vous avez choisi Cordova/PhoneGap pour vous aider dans cette voie.

Cette première partie va vous permettre tout d'abord de vous familiariser avec ces deux outils, qui ne sont pas synonymes comme vous allez le voir. Nous brosserons ensuite le tableau d'un environnement de développement idéal, en vous accompagnant dans les diverses étapes d'installation et de configuration nécessaires au bon déroulement de votre projet. Que ce soit le terminal, Node JS, les kits de développement iOS ou Android, ou encore la fameuse CLI Cordova, vous aurez ainsi toutes les cartes en mains pour vous lancer dans le développement, ce qui fera l'objet de la deuxième partie de cet ouvrage.

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

1

Cordova ou PhoneGap ?

Si vous êtes un développeur web intéressé par la création d'applications pour appareils mobiles, tels les smartphones et tablettes, il y a de grandes chances pour que vous ayez déjà entendu parler de Cordova ou PhoneGap. Peut-être vous demandez-vous alors, comme bon nombre de développeurs web, quelle est la différence entre les deux ?

Celle-ci semble à première vue assez mince, voire floue, tant certaines ressources en ligne ont tendance à mélanger ces termes. Depuis l'arrivée de Cordova, une réelle confusion s'est installée, jusque dans l'esprit des habitués de PhoneGap. Alors qu'en est-il vraiment ? Doit-on utiliser Cordova ou PhoneGap ? En réalité, vous allez voir, c'est très simple…

Un peu d'histoire

HTML 5

Le 9 janvier 2007, Steve Jobs lève le voile sur l'iPhone, ce smartphone révolutionnaire que le monde entier connaît aujourd'hui. L'appareil inclut alors quelques programmes, dont Safari Mobile, navigateur Internet extrêmement performant et innovant. Il est toutefois impossible à l'époque d'installer de nouveaux programmes. La seule solution proposée par Apple est l'ajout de *WebApps,* autrement dit de sites Internet améliorés, via Safari Mobile.

Steve Jobs lui-même vante alors les mérites d'HTML 5 *(Hypertext Markup Language version 5).* La nouvelle version du format de documents universellement reconnu sur la Toile. Avec l'HTML 5, bien que le standard ne soit pas encore totalement finalisé, les développeurs auront accès à de nombreux nouveaux outils prometteurs. L'ambition est belle : faire évoluer le Web depuis un modèle de documents vers de vrais programmes complexes.

Copyright © 2015 Eyrolles.

L'App Store et consorts

Le monde n'est semble-t-il pas encore prêt pour les WebApps, Apple pivote alors et annonce l'App Store le 6 mars 2008. Enfin, l'iPhone s'ouvre : on ne parle désormais plus de programmes, mais d'applications. L'infrastructure est inédite, le succès immédiatement et indéniablement au rendez-vous. Les développeurs peuvent maintenant créer et distribuer/monétiser « facilement » de nouvelles fonctionnalités pour le smartphone *à la pomme*. Il leur faudra juste maîtriser un langage propriétaire : l'Objective-C – extension du C.

Bien entendu, la réussite rencontrée par l'App Store fit des émules et très vite le *Google Play Store* (anciennement *Google Marketplace*) et le *Windows Phone Store* apparaissent, respectivement pour Android et Windows Phone. Notons tout de même qu'au niveau mondial les trois systèmes d'exploitation pour mobiles les plus utilisés actuellement sont Android, iOS et Windows Phone (toutefois assez loin derrière pour le moment).

Figure 1-1
Statistiques des OS mobiles pour mars 2013
Source : d'après Kantar Worldpanel ComTech

Copyright © 2015 Eyrolles.

HTML 5, le retour

S'il est nécessaire de parler à iOS en Objective-C, Android ne comprend que le Java, et Windows Phone, quant à lui, accepte idéalement le C++. C'est face à cette diversité de langages que PhoneGap a vu le jour. En réalité, le *framework* a été créé par la société Nitobi Software, lors d'un *iPhone-DevCamp* à San Francisco en 2009, dans le but de proposer aux développeurs iOS une alternative plus simple à l'Objective-C et sa syntaxe déroutante. Cette alternative n'est ni plus ni moins que l'HTML accompagné de CSS *(Cascading Style Sheets)* et JavaScript… les langages du Web.

Cette fois, l'idée de développer des applications pour iPhone en HTML plaît. Adobe rachète Nitobi fin 2011 et PhoneGap (désignant à l'époque le projet open source) est donné à la fondation Apache Software afin de demeurer un composant libre (sous licence Apache 2.0). Le projet devient alors Apache Cordova, Cordova étant le nom de la rue où sont situés à l'époque les locaux de Nitobi à Vancouver. Aujourd'hui, PhoneGap fait référence aux solutions estampillées Adobe utilisant techniquement Apache Cordova.

Cordova en bref

Figure 1-2
Le site officiel d'Apache Cordova

Pourquoi ?

Cordova permet donc de développer des applications pour iOS (c'est-à-dire destinées aux smartphones et tablettes et distribuées via l'App Store) avec les langages du Web. Celles-ci, contrairement aux WebApps traditionnelles, ont la particularité de s'exécuter dans un contexte différent leur offrant (du moins pour le moment) davantage de libertés. En effet, le futur d'HTML 5 laisse entrevoir de belles choses jusque-là encore compliquées, voire impossibles à réaliser dans un navigateur Internet. Citons par exemple la gestion d'une caméra pour la prise de photos et/ou de vidéos, ou encore l'accès au répertoire de l'appareil.

Là où HTML 5 n'est pas encore totalement présent, Cordova excelle en proposant immédiatement l'accès à des API *(Application Programming Interface)*, ailleurs encore changeantes et donc

Copyright © 2015 Eyrolles.

relativement peu implémentées par les éditeurs de navigateurs Internet. Le framework joue même la carte de la transparence, car les contributeurs du projet s'assurent de suivre scrupuleusement les spécifications W3C *(World Wide Web Consortium)* et WHATWG *(Web Hypertext Application Technology Working Group)*. Chose importante, car dans quelques années, lorsque HTML 5 sera finalisé et disponible en totalité dans nos navigateurs Internet, Cordova cessera simplement d'exister, ce qui est d'ailleurs officiellement son but à long terme.

Comment ?

Question fonctionnement, Cordova est ce que nos chers collègues anglophones appellent couramment un *wrapper* : un enrobage ou le fait d'utiliser de manière créative les outils mis à disposition par Apple afin d'en concevoir de nouveaux. En effet, le SDK *(Software Development Kit)* iOS peut facilement s'accommoder des idées les plus folles en ouvrant l'accès aux capteurs et autres fonctionnalités innovantes de l'iPhone/iPad. Et si on souhaite afficher une page HTML sur son bel écran tactile, on s'intéressera à un composant spécialement prévu pour cela : la *WebView* (plus ou moins une fenêtre de Mobile Safari dépourvue de barre d'URL et autres menus).

Sans toutefois rentrer dans le détail, Cordova exploite donc la WebView (en plein écran) en augmentant cette dernière avec certaines API HTML 5 très utiles, lesquelles se chargeront alors de communiquer avec la machine en Objective-C via JavaScript. Cordova est une sorte de traducteur JavaScript vers Objective-C (et vice versa) reposant sur les standards du Web, fonctionnant de façon extrêmement simple et quasi transparente. Une application web exécutée dans ce contexte, dialoguant avec du code natif, est nommée « application hybride ».

Pour qui ?

Un des principes fondamentaux du Web est qu'il s'agit d'une plate-forme mondiale libre, ouverte et relativement facile d'accès. C'est ce qui fait son succès, ce qui explique qu'on le retrouve aujourd'hui partout sur nos smartphones et tablettes, quels que soient leur marque et système d'exploitation. Oui, chaque appareil inclut son navigateur Internet… et par conséquent sa WebView. Peu importe que cette dernière soit écrite en Java, C++ ou Objective-C, elle aura toujours pour finalité l'affichage de fichiers HTML.

En partant de ce constat simple, l'équipe à l'origine de la création de Cordova a eu l'intelligence de ne pas s'arrêter à l'iPhone. Elle a travaillé d'arrache-pied à prendre en charge la concurrence de manière totalement transparente et cohérente. Ainsi, iOS, Android, BlackBerry, Fire OS, Ubuntu, Windows Phone, Windows 8 et Tizen sont aujourd'hui officiellement compatibles – Cordova faisant abstraction de leurs différences dans la plupart des cas, autant que faire se peut.

Remarque

Il s'agit là d'une liste de plates-formes en constante remise en cause. En effet, lorsque l'une d'entre elles n'est plus jugée comme suffisamment présente sur le marché, sa compatibilité n'est simplement plus maintenue dans les versions suivantes de Cordova.

Copyright © 2015 Eyrolles.

En pratique

Malgré la vaste panoplie de systèmes d'exploitation pris en charge par Cordova, nous ne traiterons dans cet ouvrage que d'iOS et Android, du fait de leurs parts de marché dominantes (se référer à la figure 1-1). Pour ces deux plates-formes, l'ensemble des fonctionnalités offertes par Cordova est pleinement accessible comme le montre le tableau 1-1, tiré de la documentation officielle. Citons, par exemple, la gestion du système de fichiers, l'accès aux différents capteurs présents sur les appareils, ainsi qu'à certaines données utilisateur, mais aussi l'ajout d'événements permettant d'agir à différents moments du cycle de vie des applications et de la connexion réseau.

Tableau 1-1. Fonctionnalités offertes par Cordova pour les plates-formes prises en charge

	Amazon FireOS	Android	BlackBerry 10	Firefox OS	iOS	Ubuntu	Window Phone 8	Windows 8	Tizen
CLI Cordova	✓ (Mac, Windows, Linux)	✓ (Mac, Windows, Linux)	✓ (Mac, Windows)	✓ (Mac, Windows, Linux)	✓ (Mac)	✓ (Ubuntu)	✓ (Windows)	✓	
WebView embarquée	✓	✓			✓	✓			
Ajout de plug-ins	✓	✓	✓		✓	✓	✓	✓	
Fonctionnalités officielles									
Accéléromètre	✓	✓	✓	✓	✓	✓	✓	✓	✓
Appareil photo	✓	✓	✓	✓	✓	✓	✓	✓	✓
Capture Audio/Vidéo	✓	✓	✓		✓	✓	✓		
Boussole	✓	✓	✓		✓ (3Gs+)	✓	✓	✓	✓
État de la connexion	✓	✓	✓		✓	✓	✓	✓	✓
Répertoire de contacts	✓	✓	✓	✓	✓	✓	✓		
Informations sur l'appareil	✓	✓	✓	✓	✓	✓	✓	✓	✓
Événements spécifiques	✓	✓	✓		✓	✓	✓	✓	✓
Système de fichiers	✓	✓	✓		✓	✓	✓	✓	
Géolocalisation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Internationalisation	✓	✓			✓	✓	✓		
Navigateur Internet intégré	✓	✓	✓		✓	✓	✓	✓	
Lecteur multimédia	✓	✓	✓		✓	✓	✓	✓	✓
Notifications	✓	✓	✓		✓	✓	✓	✓	✓
Écrans de lancement	✓	✓	✓		✓	✓	✓	✓	
Stockage interne	✓	✓	✓		✓	✓	✓ (LocalStorage et IndexedDB)	✓ (LocalStorage et IndexedDB)	✓

Source : Cordova

Copyright © 2015 Eyrolles.

Avec sa version 3, Cordova a subi un changement d'architecture profond. Celle-ci s'articule désormais autour d'un modèle cœur/plug-ins dans lequel chacune des API listées devient optionnelle et devra être ajoutée selon les besoins de l'application à construire. Pour l'occasion, modularité rime avec simplicité et légèreté. Toutes les API composant alors historiquement Cordova sont autant de plug-ins publiés maintenant dans un catalogue au même niveau que ceux créés par la communauté (http://plugins.cordova.io/). Nous ne détaillerons pas davantage le fonctionnement des versions antérieures, de plus amples informations à leur sujet sont disponibles dans d'autres publications, ainsi que sur la Toile.

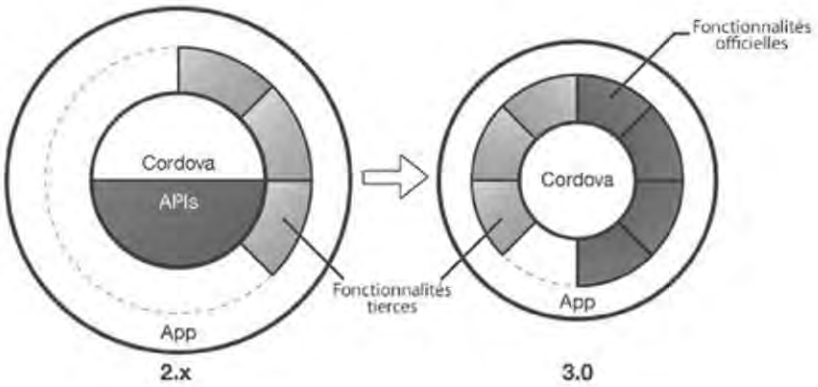


Figure 1-3
L'infrastructure de Cordova pré et post-version 3
Source : PhoneGap

Figure 1-4
Le catalogue officiel de plug-ins pour Cordova

Copyright © 2015 Eyrolles.

PhoneGap en bref

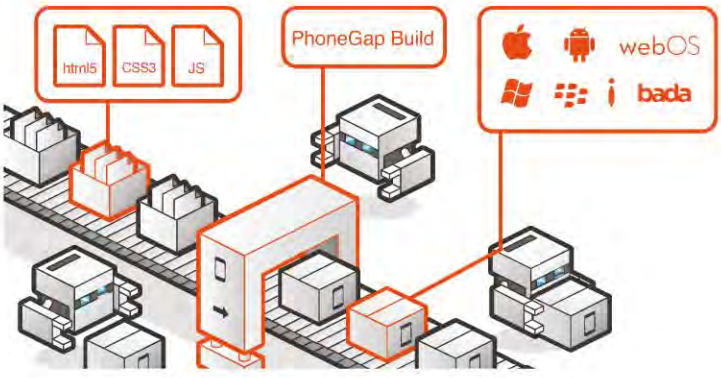
Figure 1-5
Le site officiel d'Adobe PhoneGap

Pourquoi ?

Maintenant que vous êtes familiarisé avec Cordova, il est grand temps d'aborder PhoneGap. Détenue par la société Adobe, cette marque fait désormais plus référence à de multiples services qu'à une simple technologie. Forte de sa longue expérience dans le domaine de la création graphique et web, Adobe a dans un premier temps intégré la solution à des outils de la Creative Suite tels que Dreamweaver. Puis l'entreprise a développé une série de services autour de PhoneGap, comme PhoneGap Build (https://build.phonegap.com/) et plus récemment PhoneGap Enterprise (http://enterprise.phonegap.com/), un portail des solutions PhoneGap dédiées aux entreprises (support, marketing, etc.).

Si le projet PhoneGap est simplement basé sur Cordova, PhoneGap Build est en réalité un des produits phares de la marque. Avec l'aide de Cordova, le développement d'applications pour smartphones et tablettes est déjà indéniablement simplifié, car homogénéisé. Mais PhoneGap Build va encore plus loin car ce service en ligne ne promet rien de moins que la compilation des applications PhoneGap dans le Nuage (voir figure 1-6).

Figure 1-6
Fonctionnement de PhoneGap Build
Source : PhoneGap



Copyright © 2015 Eyrolles.

Quels avantages ?

PhoneGap Build est aujourd'hui la principale réponse à la question : « Pourquoi choisir PhoneGap plutôt que Cordova ? » Cordova/PhoneGap libère les développeurs de la contrainte d'apprendre à maîtriser les langages de programmation et API propres aux différentes plates-formes. PhoneGap Build, quant à lui, les décharge de toutes les subtilités liées à la compilation des binaires destinés à être déposés sur les magasins d'applications en ligne. Pourtant, si PhoneGap Build est pour beaucoup un atout majeur, vous découvrirez au fil de cet ouvrage les avantages et inconvénients de ce service. Vous apprendrez aussi pourquoi nous préférons malgré tout bien souvent nous en passer et utiliser simplement Cordova plutôt que PhoneGap, bien que les deux outils soient interchangeables.

Enfin, le développement d'applications mobiles avec des technologies telles que HTML, CSS et JavaScript est, dans son ensemble, clairement une aubaine pour les développeurs web non particulièrement désireux d'apprendre d'autres langages.

L'idée d'un code unique pour plusieurs plates-formes est – bien que ce ne soit pas toujours réalisable – probablement le plus gros point positif de Cordova/PhoneGap. Cependant, ce type d'applications conviendra en général mieux aux projets relativement peu ambitieux, car certaines interfaces seront au final plus simples/rapides à coder en natif – natif qui fournira également toujours de meilleures performances que celles atteignables au sein d'une WebView.

Attention

Il est crucial de bien peser le pour et le contre avant de faire ce choix déterminant. Une application de montage vidéo n'est, par exemple, pas une bonne candidate pour Cordova/PhoneGap, tandis qu'un réseau social comme Twitter pourrait l'être.

En pratique

Si Adobe a accepté de donner Cordova à la fondation Apache, les contributeurs au projet restent majoritairement ses employés, aidés de développeurs travaillant pour des entreprises de renommée mondiale telles que Microsoft, IBM, Intel, Google et BlackBerry. Chacun peut également décider d'apporter son aide à titre bénévole en soumettant d'éventuels correctifs, voire en décrivant simplement tout problème rencontré sur un portail dédié dont l'accès est possible via le site officiel de l'outil.

Côté usage, d'après les quelques données statistiques publiquement disponibles, le framework Cordova/PhoneGap a été téléchargé plus d'un million de fois à ce jour par plus de 400 000 développeurs à travers le monde. Il s'agit donc bien là du leader de l'application mobile multi-plates-formes avec, en juillet 2011, plus de 500 000 visiteurs par mois sur son site officiel. Le service PhoneGap Build (payant) comptait, à la même période, plus de 10 000 utilisateurs avec un volume de 23 000 applications compilées et une croissance de 2 000 utilisateurs par mois. On peut donc facilement imaginer que ces chiffres ont doublé depuis lors.

Copyright © 2015 Eyrolles.

Remarque

PhoneGap a remporté plusieurs prix prestigieux tels que celui de la technologie de l'année 2012 et celui du meilleur outil de développement *cross-platform*.

Aujourd'hui, plusieurs milliers d'applications hybrides, réalisées à l'aide de Cordova ou PhoneGap, contribuent à la bonne santé de l'App Store, du Google Play Store et du Windows Phone Store, pour ne citer qu'eux. Quelques-unes d'entre elles sont d'ailleurs très connues, pas très loin du haut des classements dans leurs catégories. Espérons qu'elles soient vite, grâce à cet ouvrage, rejointes par les vôtres !

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

2

Prérequis et installation

En plus de demander une bonne connaissance des langages du Web (JavaScript, HTML et CSS), développer des applications mobiles avec l'aide de Cordova/PhoneGap nécessite également l'assimilation de quelques notions complémentaires relatives à certains outils tels que le terminal et Node.js. Que vous débutiez ou soyez déjà parfaitement à votre aise avec ceux-ci, ce chapitre vous montrera comment partir du bon pied pour vous préparer à l'usage des interfaces en ligne de commande Cordova et PhoneGap.

Phase « terminal »

Depuis sa version 3.0, Cordova/PhoneGap n'est plus disponible en tant qu'archive à télécharger, mais uniquement sous la forme d'interfaces en ligne de commande (plus communément dénommées CLI pour *Command-Line Interface* en anglais). Il faut les installer, puis les exécuter dans une console ou un terminal. Avant de se lancer dans l'installation à proprement parler, une mise au point basique concernant le terminal peut s'avérer nécessaire. Les développeurs déjà familiers avec cet outil peuvent passer directement à la partie II du livre.

Shell

Chaque ordinateur, au niveau logiciel, est composé d'un kernel (noyau) et de ce que l'on appelle un shell (une coquille). Ce dernier représente la couche externe via laquelle un utilisateur dialoguera avec le noyau. Il existe principalement deux sortes de coquilles connues des amateurs d'informatique : celle permettant la gestion d'un ordinateur par le biais d'une interface graphique (par exemple, le Finder sous Mac OS X, Windows Explorer, KDE ou GNOME sous Linux, etc.) et, en opposition totale, celle invitant à la saisie de commandes textuelles.

Copyright © 2015 Eyrolles.

Destinés d'abord au grand public, les systèmes d'exploitation des machines généralement vendues aujourd'hui sont prévus pour utiliser l'interface graphique par défaut, mais ceux-ci intègrent néanmoins les deux shells. Et s'il est tout à fait possible de changer ce paramètre, l'utilisateur lambda n'aura pourtant probablement aucun intérêt à effectuer cette modification. Le développeur quant à lui, soucieux de pouvoir contrôler son système en profondeur, sera naturellement intéressé par l'interface en ligne de commande qu'il utilisera pour lancer des programmes, gérer des fichiers et répertoires, et bien plus encore.

Selon leurs besoins, certains – notamment sous Linux – se passeront même totalement d'interface graphique, trouvant cette méthode globalement moins efficace. D'autres préféreront ne rien changer au fonctionnement de base de leur système d'exploitation, étant habitués à l'intuitivité des fenêtres, boutons, barres de programmes et autres menus actionnés par une souris ou un clavier. Quelle que soit votre préférence, utiliser Cordova/PhoneGap nécessite aujourd'hui l'emploi des deux méthodes. Somme toute, il s'agit là d'une excellente raison pour se familiariser avec l'interface en ligne de commande, si tel n'est pas déjà le cas !

Bash

En informatique, il existe presque toujours plusieurs outils capables de fournir le même résultat. Après tout, comme un shell n'est qu'un programme interchangeable, il est logique d'en trouver de multiples dérivés, chacun venant avec son lot de versions plus ou moins abouties, ainsi que son support pour des plates-formes données. Un des plus connus d'entre eux, par exemple, est le Bourne Shell (sh) créé par Stephen Bourne[1] dans les années 1970. Cela étant dit, d'autres appellations, comme Bash, DOS, ZSH…, devraient vous être familières, au moins de nom.

Pour cet ouvrage, il fallait faire un choix et celui-ci s'est naturellement porté sur Bash (pour *Bourne Again Shell*), shell développé vers la fin des années 1980 par Brian Fox[2] et conçu comme une évolution du Bourne Shell.

Pourquoi devoir choisir et pourquoi Bash ? principalement par souci de simplicité. En effet, comme il est possible de développer des applications Cordova/PhoneGap aussi bien sous Windows que Mac OS ou Linux, il était alors nécessaire de déterminer quel shell se révélait potentiellement compatible avec ces trois systèmes d'exploitation.

Par chance, Bash est le shell inclus par défaut dans Mac OS depuis la version 10.3 et un bon nombre de distributions Linux telles qu'Ubuntu. Les fervents utilisateurs de Linux, dont le shell par défaut n'est pas Bash, sauront très probablement soit en utiliser un autre de façon similaire, soit installer et configurer Bash si besoin. Quant aux utilisateurs de Windows, sachez qu'il existe pour vous quelques portages de Bash plus ou moins complets, mais toujours amplement suffisants dans le cadre de cet ouvrage. La version que nous conseillons est celle fournie avec Git.

1. http://fr.wikipedia.org/wiki/Stephen_Bourne
2. http://fr.wikipedia.org/wiki/Brian_Fox

Copyright © 2015 Eyrolles.

Parenthèse Git

Outre le fait d'être un outil qui – vous verrez pourquoi par la suite – est de toute façon nécessaire à tout bon développeur, Git est un programme de type CLI, donc piloté par des commandes textuelles. Excepté sous Windows, Git est parfois déjà présent dans certains des systèmes d'exploitation les plus connus. Cependant, la version embarquée est bien souvent suffisamment en retard pour que l'on souhaite tout de même installer une mise à jour. Pour ce faire, on se rendra sur le site officiel de l'outil (voir figure 2-1) afin de choisir le bon installeur. Après quoi il suffira simplement de suivre la procédure indiquée.

Figure 2-1
Téléchargement de Git sur son site officiel

Émulateur de terminal

Comme l'interface graphique est aujourd'hui la plus répandue, afin de tout de même pouvoir aisément entrer des commandes textuelles, un programme généralement nommé terminal (console, ou encore invite de commandes) est fourni dans tout système d'exploitation. Cet émulateur s'appuie sur un interpréteur de commandes donné pour transmettre des ordres au noyau dans un langage différent, compréhensible par ce dernier. De plus, un lot de commandes plus ou moins complet accompagne forcément le tout. Celles-ci, sous la forme de fichiers exécutables, sont préinstallées à divers endroits clés du système. Fort heureusement, il est inutile d'atteindre à chaque fois ces emplacements pour exécuter les commandes souhaitées.

En effet, Bash et équivalents ont pour instruction de suivre les chemins usuels indiqués dans une variable d'environnement (c'est-à-dire définie au niveau système et donc partagée entre plusieurs programmes) savamment nommée `PATH`, de l'anglais « chemin ». Dans la pratique, l'utilisateur n'entrera alors que le nom de la commande à exécuter, puis Bash parcourra les répertoires indiqués dans le `PATH` à la recherche de la première correspondance si tant est bien sûr qu'il y en ait une. Sans pour autant entrer ici davantage dans le détail, sachez que le `PATH` est facilement personnalisable. On pourra, par exemple, ajouter des chemins ou bien encore modifier l'ordre de ceux-ci afin de cibler en premier certains exécutables présents plusieurs fois sur le système, mais dans des versions différentes.

Copyright © 2015 Eyrolles.

Avec ces quelques notions en tête, il est grand temps d'ouvrir une fenêtre de terminal et d'entrer ce qui sera peut-être votre toute première commande. On démarrera pour cela `Terminal.app` sous Mac OS X, l'utilitaire Git Bash (fraîchement installé) sous Windows et la console de base sous Linux. Mises à part quelques couleurs et autres légères différences stylistiques selon les plates-formes et logiciels, vous devriez avoir devant vous une fenêtre plutôt vide dans laquelle est affiché le symbole bien connu du dollar suivi d'un curseur clignotant, invitant à la saisie de texte.

Remarque

Il existe différentes consoles permettant chacune plus ou moins de réglages et autres options de confort. On retiendra notamment iTerm (http://www.iterm2.com/) sous Mac et Console 2, une alternative pour Windows (http://sourceforge.net/projects/console/) ; logiciels tous deux très populaires auprès des développeurs.

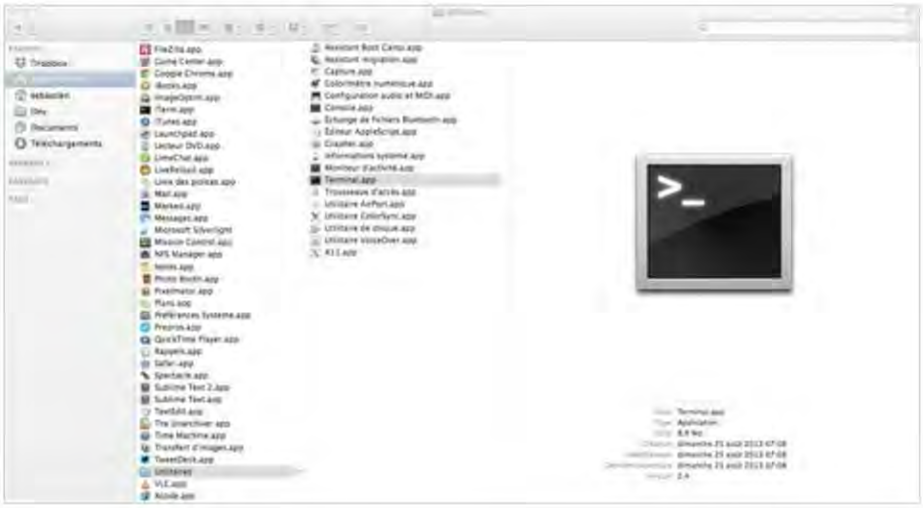


Figure 2-2
Terminal.app sous Mac OS X Mavericks

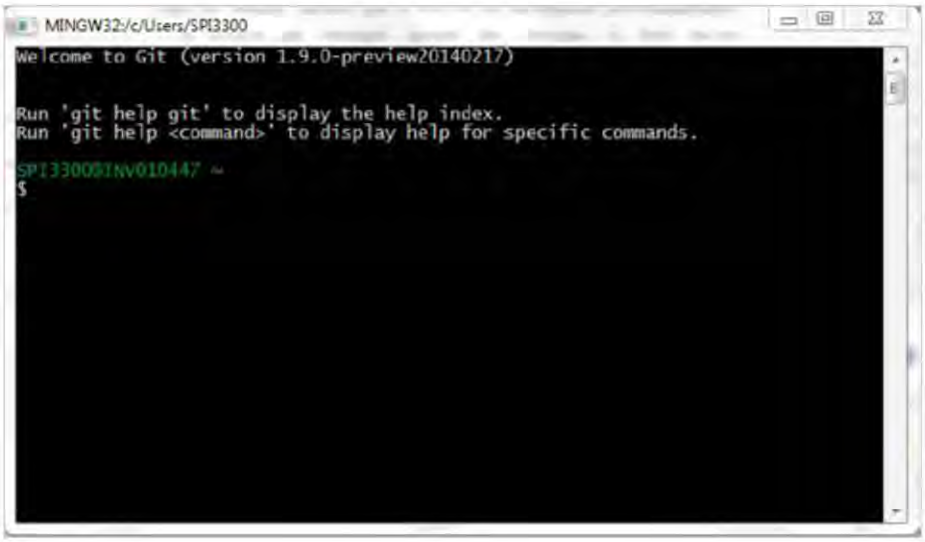


Figure 2-3
Git Bash sous Windows 7

Copyright © 2015 Eyrolles.

Notez que le curseur en question peut être précédé d'un dollar, ou bien de presque n'importe quel symbole, lui-même possiblement accompagné d'autres caractères correspondant à ce que l'on nomme le prompt. Il s'agit là d'un message également personnalisable le plus souvent utilisé pour afficher au moins l'adresse du répertoire dans lequel on se situe. Oui, car l'effet de certaines commandes, vous le verrez par la suite, dépend de leur contexte d'exécution. Si votre prompt est prévu pour afficher cette information, sachez que ~ (tilde) fait par convention référence à la racine de votre répertoire utilisateur et / (slash) à celle du système de fichiers.

Entrer des commandes

Sans rien écrire, appuyez plusieurs fois sur la touche Entrée de votre clavier. Rassurez-vous, ceci n'a aucun autre effet que de sauter des lignes. Entrez maintenant `clear`, puis appuyez de nouveau sur la touche Entrée. Vous remarquerez alors que toutes les lignes inutiles précédemment ajoutées à l'historique ont été retirées de l'affichage. Mieux encore, vous venez d'utiliser la commande `clear` dont tel est précisément le but !

Figure 2-4
Utilisation de la commande clear

Saisissez maintenant la commande `pwd` *(print working directory)* ; vous pourrez ainsi vérifier dans quel dossier vous vous trouvez. Changer de répertoire est chose aisée grâce à la commande `cd` *(change directory)*, entrez `cd` pour aller dans votre répertoire utilisateur. Si vous souhaitez lister son contenu, entrez `ls`. Remarquez que tapez `ls` de cette façon présente les données en plusieurs colonnes et n'affiche pas, entres autres, les fichiers cachés (c'est-à-dire, par convention sous Unix, ceux dont le nom commence par un point). Essayez alors `ls -lA`, c'est un peu mieux et vous découvrez par ailleurs deux des multiples options qu'offre `ls`.

Arguments et options

Bien qu'il existe différentes conventions, et que celles-ci ne soient pas toujours entièrement respectées par les développeurs, une commande est sensible à la casse et tient en général sur une ligne. Son nom suit directement le prompt, après quoi viennent d'éventuels groupes de caractères séparés par un espace. On parle alors d'arguments et d'options destinés à modifier le comportement par défaut d'une commande. Par exemple, `ls ../..` affiche le contenu du répertoire situé deux niveaux au-dessus du répertoire de travail sans avoir à exécuter successivement `cd ../..` et `ls`. Dans ce cas, `../..` est un argument. Notez que tout espace constituant un argument doit logiquement être échappé, soit en insérant un \ (antislash) juste avant, soit en mettant l'argument entier entre guillemets.

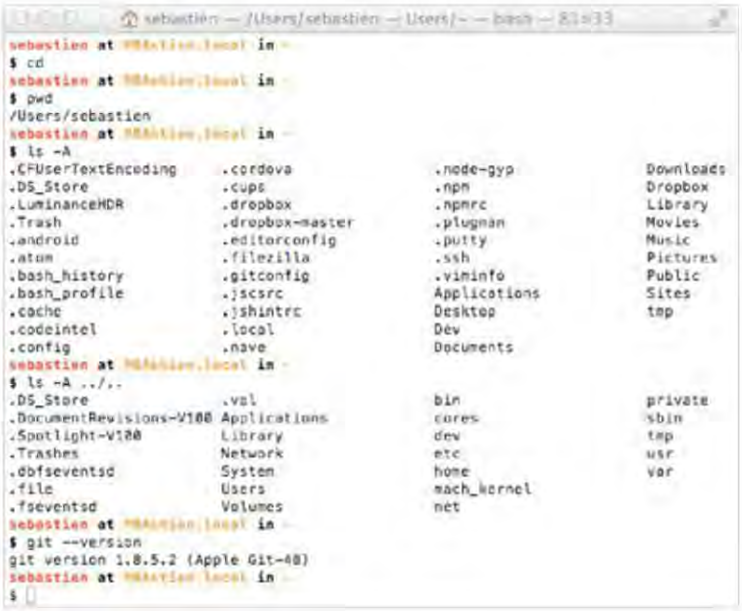
Copyright © 2015 Eyrolles.

Les options ont, quant à elles, une syntaxe légèrement plus complexe. Prenons un exemple fictif : `command --option1 --option2`. Il s'agit là d'une commande possédant au moins deux options, toutes deux utilisées en même temps à l'exécution. Le nom de chaque option est précédé de deux tirets par convention Unix. Cette syntaxe est nommée « version longue ». Saisir régulièrement de telles commandes peut s'avérer fastidieux, c'est pourquoi une forme abrégée existe. Cette dernière est, bien que forcément moins parlante, préférée des développeurs et parfois même la seule existante (c'est notamment le cas pour `ls`) – l'inverse peut aussi arriver. On pourra alors écrire `command -1 -2` ou mieux `command -12`. Vérifions, par exemple, la présence de Git sur le système en entrant `git --version` (la forme abrégée n'étant pas disponible).

Astuce

Sous Bash, il est possible de créer des raccourcis (http://doc.ubuntu-fr.org/alias) afin de réduire l'effort de frappe demandé pour exécuter des commandes assez longues et fréquemment utilisées. De même, les flèches haut et bas du clavier permettent de naviguer dans l'historique du terminal. Pratique pour éviter de taper plusieurs fois une même commande.

Figure 2-5
Exemples de commandes avancées



Aide et manuel d'utilisation

Évidemment, si la commande utilisée le permet, arguments et options pourront être combinés. Selon les cas, on précisera d'abord les options, puis les arguments, ou bien chaque option précédera un argument… difficile à prévoir. Avant tout, l'usage de chaque commande nécessite une connaissance particulière de son but, ainsi que de son fonctionnement. Fort heureusement, ces outils sont, pour la plupart, livrés avec un manuel d'utilisation complet accessible directement depuis le terminal ou bien au minimum consultable en ligne. Là encore, suivant l'implémentation des différentes commandes, l'accès à une aide ne se fait pas systématiquement de la même façon.

Copyright © 2015 Eyrolles.

Voici certains des moyens d'accès les plus souvent rencontrés :

- `command` ①
- `command -?` ②
- `command -h` ③
- `command -H` ④
- `command --help` ⑤

Entrer seulement le nom de la commande n'est pourtant pas recommandé, car certaines exécuteront alors simplement leur comportement par défaut, tandis qu'ajouter une option inconnue provoquera une erreur. C'est notamment le cas de `cd` et `ls` qui, quant à elles, ne sont pas dangereuses, car non destructrices. De même, toujours concernant `ls`, ③ et ④ n'afficheront pas d'aide, car les options `-h` et `-H` auront ici une tout autre signification. Alors, comment faire ? Utiliser la commande `man`[3] (pour *manual*). Entrez `man ls` (`man` suivi du nom de la commande souhaitée, précédée d'un espace). Vous devriez maintenant avoir à l'écran la notice d'utilisation complète de la commande `ls` tel qu'illustrée par la figure 2-6. On se servira des flèches haut et bas du clavier pour naviguer et de la touche « q » pour quitter en temps voulu.

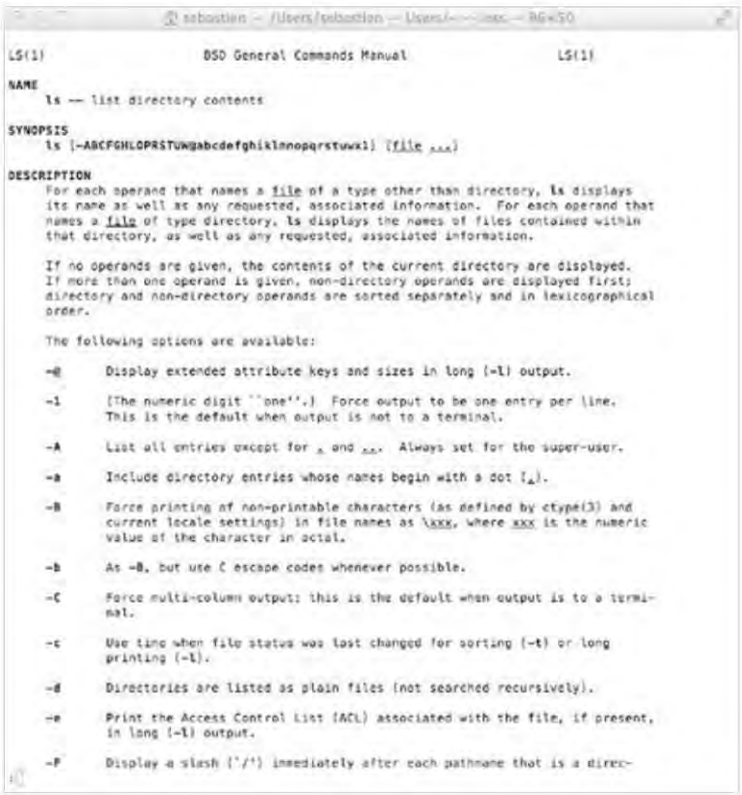


Figure 2-6
Manuel d'utilisation de la commande ls

3. http://fr.wikipedia.org/wiki/Man_(Unix)

Copyright © 2015 Eyrolles.

Malheureusement, toutes les commandes ne possèdent pas de pages `man`. Si aucune des méthodes listées précédemment ne donne de résultats, il ne vous reste plus qu'à chercher sur Internet. Pour vous faire la main, consultez les manuels des commandes :

- `mkdir` *(make directory)* pour créer un répertoire ;
- `rm` *(remove)* pour supprimer des répertoires et fichiers ;
- `cp` *(copy)* pour dupliquer des répertoires et fichiers ;
- `mv` *(move)* pour renommer/déplacer des répertoires et fichiers ;
- `touch` pour créer des fichiers ;
- `chown` *(change owner)* pour modifier les droits d'accès aux répertoires et fichiers ;
- `cat` pour afficher le contenu d'un fichier.

Elles pourront parfois s'avérer très utiles en complément de votre interface graphique préférée.

Anatomie d'une commande

Que ce soit dans les manuels d'utilisation ou bien dans l'aide en ligne, les développeurs de CLI se sont mis d'accord sur une norme qu'il est bon de connaître. Celle-ci vise à décrire, par exemple, l'usage de leurs outils. Ainsi, outre la connaissance nécessaire des noms des options eux-mêmes, il est toujours relativement aisé de savoir comment utiliser une commande en suivant les indications. On retiendra donc que :

- les paramètres indiqués entre crochets sont optionnels, par exemple `ls [directory]` pour `ls` ou `ls ../..` ;
- ceux entre chevrons sont au contraire requis, par exemple `rm <file>` pour `rm app.js` ;
- des points de suspension indiquent une répétition, par exemple `mkdir <directory…>` pour `mkdir dir1` ou `mkdir dir1 dir2 dirN` ;
- les choix sont symbolisés par un pipe (barre verticale), par exemple `mv [-f | -i | -n][-v] <source…>`. En effet, les options `-f`, `-i` et `-n` sont incompatibles, elles ne peuvent donc pas être utilisées simultanément.

Crochets, chevrons, points de suspension et barres verticales ne sont bien évidemment pas à conserver au moment d'entrer des commandes. Certains de ces caractères peuvent d'ailleurs avoir une signification bien particulière dans un terminal, mais ceci relève d'un usage avancé sortant du cadre de cet ouvrage.

sudo

Sous Mac OS X et Linux, pour des raisons de sécurité, lorsqu'une commande cherche à travailler dans un contexte sensible – comme un répertoire système dont les droits d'accès sont restreints –, celle-ci échoue souvent simplement (retourner une erreur quelconque), voire ne fonctionne pas correctement. Dans ce cas, si l'on est sûr de soi, il faudra l'exécuter en tant que superutilisateur afin de lui donner l'accréditation nécessaire pour pouvoir travailler comme bon lui semble. Rassurez-vous, c'est très simple, il suffit d'utiliser la commande `sudo` (pour *substitute user do)* qui, comme `man`, préfixe la commande à exécuter. Vous serez alors invité à entrer le mot de passe du

Copyright © 2015 Eyrolles.

compte administrateur, après quoi celui-ci ne vous sera plus demandé pendant un certain laps de temps. Pour plus d'informations, entrez `man sudo`.

Vous possédez maintenant toutes les connaissances nécessaires à l'utilisation d'outils en ligne de commande, il est grand temps d'aborder Node.js.

Node.js

Nous l'avons vu, depuis sa version 3.0, Cordova/PhoneGap n'est disponible à l'installation que sous la forme d'interfaces en ligne de commande et celles-ci sont en réalité des modules pour Node.js. Il est donc, avant toute chose, nécessaire de savoir ce qu'est Node.js, puis d'installer l'outil sur sa machine.

Figure 2-7
Le site officiel de Node.js

Présentation

Créé à l'origine en 2009 par Ryan Dahl[4], Node.js est un projet ambitieux, aujourd'hui maintenu par l'entreprise Joyent et la communauté open source sur GitHub. Typiquement, Node.js aide à la création de toutes sortes d'applications serveur en JavaScript. Vous avez bien lu, du JavaScript côté serveur ! Pour cela, le projet s'appuie notamment sur V8 : le moteur d'exécution JavaScript signé Google (principalement connu pour être utilisé dans le navigateur Chrome).

Tout comme Cordova/PhoneGap, Node.js offre ainsi aux développeurs un nombre conséquent d'API très pratiques permettant notamment de gérer un système de fichiers, de créer des serveurs HTTP(S), de démarrer des processus, etc. Enfin, le concept repose sur une architecture modulaire fortement asynchrone faisant intervenir événements et entrées/sorties non bloquantes. De ce fait, Node.js est théoriquement capable d'offrir d'excellentes performances, et ce, même à très grande échelle.

4. http://fr.wikipedia.org/wiki/Node.js

Copyright © 2015 Eyrolles.

De grandes sociétés telles que Microsoft, Yahoo!, LinkedIn et autres l'utilisent d'ailleurs avec succès en production depuis déjà un certain temps. Et le choix du JavaScript comme unique langage de programmation s'inscrit particulièrement bien dans l'univers du Web – ce qui séduit aujourd'hui de plus en plus de développeurs. Le projet a en effet connu un succès fulgurant, se hissant très rapidement vers le haut du classement des projets les plus suivis sur GitHub.

Installation de Node.js

Node.js est disponible en simple téléchargement (pour Mac OS X, Windows et Linux) sur son site officiel : http://nodejs.org/download/. L'installer est donc extrêmement aisé et à la portée de tous. Il suffira de récupérer l'installateur de la dernière version stable disponible pour la plate-forme de votre choix, d'exécuter ce dernier comme à l'accoutumée, puis de suivre les instructions affichées à l'écran (un redémarrage de l'ordinateur peut cependant être nécessaire sous Windows).

Figure 2-8
Téléchargement de l'installateur Node.js

Ajouter Node.js de cette façon écrasera inévitablement toute autre installation préalable de l'outil. C'est important de le souligner, car le projet étant encore relativement jeune, certaines de ses API sont amenées à évoluer entre les différentes versions de l'exécutable, pouvant ainsi provoquer des incompatibilités graves avec des modules/programmes précédemment installés et/ou créés. Pour vérifier si Node.js est déjà installé sur votre machine, ouvrez simplement une fenêtre de terminal et entrez la commande :

```
node -version
```

Vous obtiendrez alors soit une erreur précisant que la commande `node` est introuvable, soit quelque chose comme `v0.10.30`, typiquement la version de Node.js installée. Une commande introuvable peut toutefois être présente sur la machine ; en effet, son chemin d'accès ne figure pas forcément dans la variable d'environnement `PATH` (voir chapitre 1). Par défaut, Node.js se cache dans `/usr/local/bin` sous Mac et Linux et dans `C:\Program Files (x86)\nodejs\` sous Windows (voir figure 2-9).

Copyright © 2015 Eyrolles.

Figure 2-9
Installation de Node.js sous Windows 7

Plusieurs installations en parallèle

Afin d'éviter d'éventuels problèmes de compatibilité évoqués précédemment, il peut être intéressant d'avoir la possibilité de gérer plusieurs versions de Node.js en parallèle. C'est-à-dire de pouvoir aisément basculer d'une version donnée vers une autre. Pour cela, il existe d'ores et déjà plusieurs outils dont nous ne retiendrons que les plus populaires : n (https://github.com/visionmedia/n), nvm (https://github.com/creationix/nvm) et nave (https://github.com/isaacs/nave). Cependant, ceux-ci ne sont pas compatibles avec Windows, mais il existe heureusement nodist (https://github.com/marcelklehr/nodist) et nvmw (https://github.com/hakobera/nvmw), tous deux dédiés au système d'exploitation signé Microsoft. Leur fonctionnement est extrêmement proche de n, nvm et nave, puisque leurs auteurs s'en sont directement inspirés (se référer à leurs documentations respectives).

Dans le but de faciliter l'installation de tels scripts, Node.js est normalement livré avec un gestionnaire de paquets (plus généralement appelés modules) nommé npm (https://github.com/isaacs/npm) pour *Node Package Manager*. npm n'est lui-même rien d'autre qu'un module pour Node.js servant à installer d'autres modules, ainsi que leurs dépendances. Sa présence est toutefois cruciale tellement son intérêt est grand. Pour vérifier si npm est installé sur votre système, ouvrez simplement une fenêtre de terminal et entrez la commande :

```
npm -version
```

Pas de surprise ici, vous obtiendrez encore une fois soit une erreur, soit le numéro de version de npm. Si npm n'est pas installé, mais que Node.js est bien présent, il est fort probable qu'il s'agisse d'une très ancienne version de Node.js ou bien que npm ait été supprimé. Dans ce cas, reportez-vous aux explications présentes sur le dépôt GitHub de npm, ou bien suivez les instructions suivantes pour réinstaller l'ensemble.

Parmi les trois gestionnaires de version de Node.js cités ci-dessus, nous choisirons nave pour deux raisons : sa simplicité générale, et parce que son auteur est également celui de npm – un gage de qualité.

Copyright © 2015 Eyrolles.

Nous allons donc une fois de plus tâcher de déterminer si nave est déjà installé en entrant la commande :

```
nave -version
```

Si tel est le cas, assurons-nous d'avoir la version la plus récente disponible :

```
npm update --global nave
```

Sinon, procédons simplement à son installation :

```
npm install --global nave
```

Attention

Les modules pour Node.js sont par défaut installés localement dans un répertoire `./node_modules` créé pour chaque projet. Certains, notamment ceux correspondant à un outil en ligne de commande, nécessitent en général une installation globale – c'est-à-dire proche du répertoire d'installation de Node.js – afin d'être placés dans le `PATH` et donc directement exécutables.

Pour rappel, les procédures d'installation et de mise à jour de paquets via npm nécessitent évidemment une connexion à Internet. Par conséquent, la durée de téléchargement des composants nécessaires dépendra notamment du débit disponible. Veillez à bien attendre la fin du processus. De même, excepté sous Windows, il est possible que vous n'ayez pas les droits nécessaires pour effectuer ces actions. Dans ce cas, pour éviter toute erreur, il suffira de préfixer la commande souhaitée par `sudo` de la façon suivante (et d'entrer le mot de passe administrateur avant de valider) :

```
sudo npm install --global nave
```

Enfin, nous utiliserons nave pour gérer différentes versions de Node.js et npm sur une même machine. Cette étape n'est pas nécessaire si aucune version de Node.js n'était préalablement installée sur votre machine. Un utilisateur averti pourra tout à fait installer Node.js directement via nave après avoir installé le script Bash fourni, mais ceci sort du cadre de cet ouvrage. Ouvrez une nouvelle fenêtre de terminal et entrez la commande :

```
nave use cordovabook stable
```

Cela aura pour effet de télécharger la version stable la plus récente disponible de Node.js et d'utiliser temporairement celle-ci dans la fenêtre de terminal en cours d'utilisation. Autrement dit, toute installation préalable de Node.js ne sera pas impactée. Un nouvel environnement virtuel nommé `cordovabook` sera créé, telle une étiquette référençant la version de Node.js installée à l'instant. Vous devriez avoir voir ceci à l'écran :

```
Creating new env named'cordovabook' using node 0.10.30
######################################################################## 100,0%
installed from binary
```

Ainsi, afficher la version de Node.js utilisée dans cette fenêtre de terminal devrait renvoyer `0.10.30`. Faire la même chose dans une nouvelle fenêtre de terminal devrait retourner une valeur

Copyright © 2015 Eyrolles.

différente correspondant à la version de votre installation préalable de Node.js. Par conséquent, afin de toujours bien exécuter les commandes `cordova` et `phonegap` dans le bon contexte, tel que décrit dans la suite de ce livre, pensez à entrer la commande suivante (ou son équivalent) à l'ouverture de chaque nouvelle fenêtre de terminal :

```
nave use cordovabook
```

Kits de développement

Parmi les dépendances nécessaires à l'installation des CLI Cordova et PhoneGap, vous possédez maintenant Git et Node.js. C'est là un excellent début, mais il vous manque encore le plus important : les différents kits de développement correspondant aux plates-formes pour lesquelles vous souhaitez développer des applications. Il s'agit ni plus ni moins de plusieurs collections d'outils mis à disposition par les grands acteurs de l'industrie afin de faciliter l'accès des développeurs à leurs systèmes d'exploitation mobiles. Rassurez-vous, la marche à suivre est cette fois encore relativement simple et rapide.

Cet ouvrage se veut non exhaustif et – bien que Cordova et PhoneGap peuvent être utilisés pour créer des applications pour FireOS (Amazon), Android (Google), BlackBerry 10 (RIM), Firefox OS (Mozilla), iOS (Apple), Ubuntu (Canonical), Tizen (Intel et Samsung), Windows Phone 8 et Windows 8 (Microsoft) – traite seulement des deux plates-formes mobiles les plus répandues aujourd'hui, à savoir Android et iOS. Nous vous invitons à consulter la documentation officielle pour davantage d'informations. Celle-ci est d'ailleurs disponible en français.

iOS

Autant vous prévenir tout de suite, à moins d'employer les bons et loyaux services de PhoneGap Build (vous verrez comment par la suite), pour développer des applications pour iOS, vous aurez besoin d'un ordinateur Apple. En réalité, vous pourrez toujours mettre en place vos applications et les tester sous n'importe quel OS dans un navigateur web, mais il vous faudra forcément un Mac pour les compiler en vue d'une distribution sur l'App Store. Le Mac Mini (https://www.apple.com/fr/mac-mini/) est par exemple un excellent moyen de débuter pour qui possède déjà un écran accompagné d'une souris et d'un clavier. On en trouve parfois à des prix très intéressants dans le programme d'appareils reconditionnés[5] Apple.

Notez qu'il est toujours préférable de posséder une machine relativement récente, car capable d'accueillir les dernières mises à jour de Mac OS X (gratuites depuis Mavericks) afin d'accéder aux dernières versions d'Xcode. Qu'est-ce qu'Xcode ? Eh bien, simplement le kit de développement officiel iOS. Pour vous le procurer, direction le Mac App Store dans la catégorie Développeurs. Le téléchargement est gratuit et ne requiert qu'un simple compte Apple (accès basique aux applications de l'App Store). Suivant le débit de votre connexion Internet, celui-ci prendra un certain temps. En effet, la suite d'outils fournie est très complète et permet beaucoup plus que le simple développement iOS.

5. http://store.apple.com/fr/browse/home/specialdeals/mac/mac_mini

Copyright © 2015 Eyrolles.

Figure 2-10
Xcode dans le Mac App Store sous Mac OS X Mavericks

L'ensemble contient un environnement de travail complet, de l'éditeur de code au simulateur iOS, en passant par de la documentation et autres frameworks et bibliothèques utiles. Une fois le téléchargement terminé, installez et lancez Xcode afin de vous rendre dans le menu Préférences sous l'onglet Locations pour vérifier la présence des Command Line Tools – à installer également le cas échéant. Ceux-ci autoriseront Cordova/PhoneGap à piloter Xcode à distance et ainsi à s'en abstraire totalement. Bien sûr, vous pourrez tout de même décider d'en faire votre éditeur de code préféré, mais toute alternative telle que Sublime Text (http://www.sublimetext.com/), WebStorm (http://www.jetbrains.com/webstorm/), Atom (https://atom.io/), Brackets (http://brackets.io/) ou autre fera aussi bien l'affaire, selon les goûts de chacun.

Figure 2-11
Accéder aux préférences d'Xcode

Copyright © 2015 Eyrolles.

Figure 2-12
Vérifier la présence des outils en ligne de commande dans Xcode 5.1.1

Le simulateur iOS accompagnant la suite Xcode est un outil extrêmement bien conçu et simple d'utilisation permettant notamment d'essayer ses applications sous plusieurs versions d'iOS, mais surtout dans le cadre de différentes tailles d'écrans. Il est donc possible de s'affranchir – même si cela est relativement peu conseillé – de posséder plusieurs appareils Apple dans les gammes iPhone, iTouch et iPad, car le rendu est en général plutôt bon et fidèle à la réalité. Placé dans le répertoire d'installation d'Xcode (sous `Contents/Developer/Platforms/iPhoneSimulator.platform/Developer/Applications` pour la version 5.1.1), `iOS Simulateur.app` mérite une petite place dans le dock ou toute autre forme de raccourci visant à simplifier son accès dans le futur.

Enfin, notez qu'au départ seule la version la plus récente d'iOS pour le simulateur est installée en même temps qu'Xcode. Bien que les produits Apple profitent toujours de mises à jour assez longtemps après leur sortie sur le marché – ceci conduisant à une adoption en masse et généralement rapide par leurs utilisateurs –, il peut être intéressant de conduire des tests de vos applications dans des versions antérieures. Vous devez alors retourner dans les préférences d'Xcode, cette fois-ci sous l'onglet Downloads, et effectuer les téléchargements souhaités dans la zone Components. Vous verrez plus tard dans cet ouvrage comment utiliser le simulateur au mieux de ses capacités.

Figure 2-13
Installer différentes versions d'iOS pour le simulateur

Copyright © 2015 Eyrolles.

Android

À la différence d'iOS, compiler des applications pour Android peut se faire aussi bien sous Windows, que Mac OS ou Linux. Mis à disposition par Google sur un portail web dédié (voir figure 2-14), le kit de développement Android est proposé en deux versions distinctes. En effet, le développement natif Android s'effectuant en Java, les développeurs habitués à ce langage s'orienteront naturellement vers Eclipse, un environnement de développement complet leur étant destiné. Ceux-ci choisiront le paquet nommé Eclipse ADT *(Android Developer Tools)* intégrant donc Eclipse préconfiguré pour le développement Android, ainsi que l'ensemble des outils nécessaires dont un émulateur. Il s'agit là de l'équivalent d'Xcode pour iOS.

Même si Java ne fait pas partie des langages nécessaires à la réalisation d'applications Cordova/PhoneGap, vous pourrez toujours vous servir d'Eclipse s'il est votre environnement de développement préféré. Sinon, nous vous conseillons de ne télécharger que la suite d'outils composant réellement le SDK Android, à savoir les Stand-alone SDK tools – plus ou moins l'équivalent des outils en ligne de commande pour Xcode. De cette façon, vous pourrez alors choisir de travailler dans n'importe quel éditeur de code pour un résultat final de toute façon identique.

Note

Google travaille actuellement sur un environnement de développement basé sur Intellij. Nommé Android Studio et disponible en version bêta, il est destiné, à terme, à remplacer l'Eclipse ADT.

Figure 2-14
Choix du paquet à télécharger pour l'installation du SDK Android

Faites votre choix et suivez les instructions pour débuter le téléchargement. Puis débutez l'installation légèrement différente sous Windows, Mac OS et Linux (un installateur ou bien des archives à décompresser). Quel que soit le système d'exploitation sur lequel vous travaillez, prenez soin de retenir le répertoire de destination des fichiers, car celui-ci vous sera utile par la suite. Choisissez d'ailleurs si possible un emplacement spécifique, comme sous `~/dev/` ou `~/tools/`, car une fois la configuration à suivre effectuée, le chemin choisi ne devra pas être amené à changer, sous peine d'empêcher tout bonnement le fonctionnement des outils.

Copyright © 2015 Eyrolles.

En effet, nous allons maintenant ajouter certains chemins importants au `PATH`. Vous comprenez donc pourquoi déplacer les fichiers et/ou répertoires concernés mènera inévitablement à des erreurs, à moins de réadapter ensuite le `PATH`. Ouvrez une nouvelle fenêtre de terminal, puis entrez :

```
cd ~ et cat .bash_profile
```

Si vous obtenez une erreur telle que :

```
cat: .bash_profile: No such file or directory
```

c'est simplement que le fichier en question n'existe pas encore sur votre système. Créez-le en entrant la commande suivante :

```
touch .bash_profile
```

Note

Créer ce fichier via le terminal permet d'outrepasser la limitation de l'explorateur Windows empêchant les noms de fichiers de commencer par un point (convention désignant les fichiers cachés sous Unix).

Vous possédez maintenant un fichier nommé `.bash_profile` situé à la racine de votre répertoire utilisateur. Il s'agit d'un fichier de configuration de Bash destiné entre autres à accueillir d'éventuelles modifications au `PATH`, cela tombe bien. Ouvrez-le alors avec votre éditeur de texte préféré (rappel, la commande `pwd` affiche le chemin vers le répertoire dans lequel vous travaillez). Si votre `.bash_profile` n'est pas vide, ne prêtez pas attention à son contenu et ajoutez simplement les lignes suivantes à sa toute fin en remplaçant `~/dev/` par le chemin d'installation du SDK précédemment mémorisé.

```
ADT_SDK=~/dev/adt/sdk
export PATH=$PATH:$ADT_SDK/platform-tools:$ADT_SDK/tools
```

Par défaut, la première ligne deviendrait par exemple sous Windows :

```
ADT_SDK="~/AppData/Local/Android/android-sdk"
```

La seconde, assez simple à comprendre, a pour effet d'ajouter les répertoires `platform-tools` et `tools` au `PATH`. Une fois la modification effectuée, sauvegardez, puis retournez dans le terminal (toujours dans `~`) et entrez `. .bash_profile` afin d'appliquer les changements. Cette dernière commande, bien qu'à première vue étrange, permet simplement d'éviter d'avoir à fermer puis rouvrir le terminal. Gardez également le fichier `.bash_profile` ouvert dans un coin de l'écran, nous allons très rapidement y revenir. Mais avant ceci, il vous faut installer encore deux outils : Java et Ant.

Note

Sous Windows, il existe une autre façon de modifier le `PATH`. La procédure décrite à la figure 2-15 est celle sous Windows 7. Rendez-vous dans Panneau de configuration>Système et sécurité>Système puis cliquez sur Paramètres système avancés, et enfin sur le bouton Variables d'environnement pour afficher l'interface dédiée. Vous pourrez ensuite créer ou modifier des variables telles que le `PATH`. Petite précision : on écrira par exemple `%ADT_SDK%` au lieu de `$ADT_SDK`, un antislash remplacera chaque slash et le caractère délimitant chaque chemin sera ici un point-virgule.

Copyright © 2015 Eyrolles.

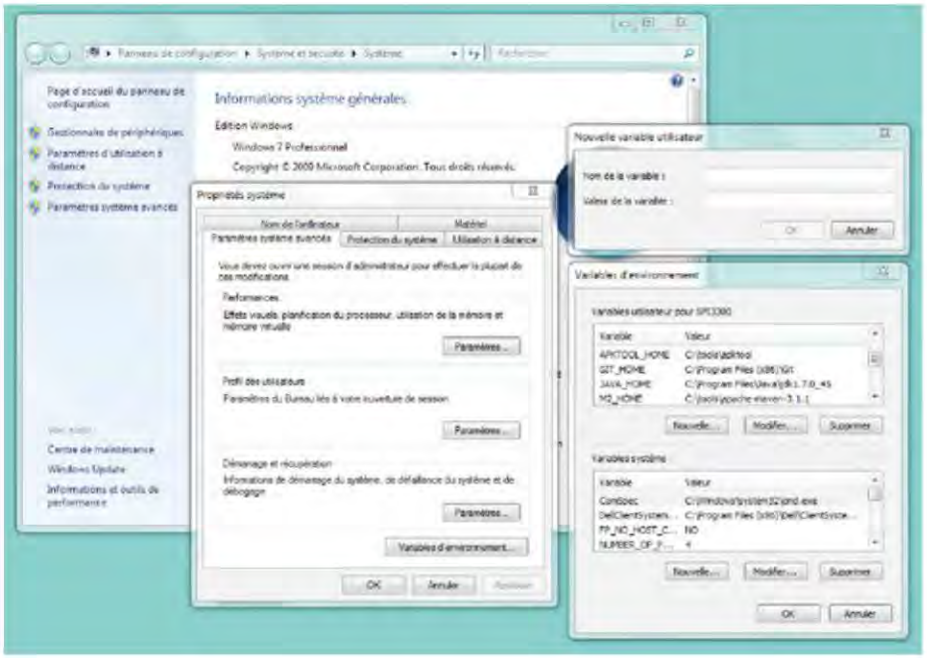


Figure 2-15
Modifier le PATH sous Windows 7

Java

L'un d'eux est Java, ou plus précisément le kit de développement pour Java comprenant un environnement d'exécution, ainsi que différents outils de debug, monitoring et autres. Google précise qu'il est nécessaire de posséder l'ensemble du kit de développement (JDK) et non pas uniquement l'environnement d'exécution (JRE). La version 6 est celle listée dans les prérequis, mais la version 7 est également parfaitement compatible. Tout comme Git, certains systèmes d'exploitation sont livrés avec Java, entrez alors la commande qui suit pour vérifier son éventuelle présence sur votre machine :

```
java -version && javac -version
```

Il s'agit en réalité de deux commandes en une, la seconde n'étant pas exécutée si la première venait à échouer.

Figure 2-16
Vérification de la présence du JDK et de sa version

Copyright © 2015 Eyrolles.

Si, comme sur la figure 2-16, vous n'obtenez aucune erreur de type `command not found` et que les versions retournées commencent par 1.6 ou 1.7, vous êtes parmi les plus chanceux. Dans le cas contraire, rendez-vous sur le site officiel d'Oracle (voir figure 2-17) afin de télécharger le paquet correspondant à votre système d'exploitation. Mêmes consignes que pour le SDK Android, suivez les instructions et choisissez un emplacement d'installation spécifique que vous retiendrez pour pouvoir aisément éditer le `PATH` en conséquence. Après installation, on modifiera alors les lignes précédemment ajoutées dans le fichier `.bash_profile` de la façon suivante, sans oublier d'ajuster la valeur de la variable `JAVA_HOME` à votre convenance :

```
ADT_SDK=~/dev/adt/sdk
export JAVA_HOME=~/dev/jdk
export PATH=$PATH:$ADT_SDK/platform-tools:$ADT_SDK/tools:$JAVA_HOME/bin
```

En guise de vérification, exécutez une nouvelle fois :

```
. .bash_profile, puis java -version && javac -version
```

Note

Dans le cadre d'une mise à jour, si vous souhaitez tout de même conserver la version précédente, choisissez un répertoire d'installation différent et remplacez la dernière ligne par :

```
export PATH=$JAVA_HOME/bin:$PATH:$ADT_SDK/platform-tools:$ADT_SDK/tools
```

Bash trouvera ainsi votre version fraîchement installée avant celle déjà présente.

Figure 2-17
Téléchargement de Java 7

Copyright © 2015 Eyrolles.

Figure 2-18
Téléchargement du JDK 7

Figure 2-19
Installation du JDK 7 sous Mac OS X Mavericks

Ant

`Ant` pour *Another Neat Tool* (que l'on pourrait traduire par « Un autre outil chouette »), est un programme de la fondation Apache dont le but est d'automatiser les tâches répétitives dans le développement de logiciels. C'est aussi le second et dernier outil requis par le SDK Android. Il assurera, par exemple, la compilation, l'optimisation et le déploiement des applications, mais rassurez-vous, son utilisation se fera par le biais des interfaces en ligne de commande PhoneGap et Cordova. Nul besoin d'apprendre les fondements de l'outil, nous nous contenterons alors d'installer celui-ci. Vous pouvez toujours essayer d'entrer la commande `ant -version`, mais il y a tout

Copyright © 2015 Eyrolles.

de même peu de chances qu'Ant soit déjà présent sur votre machine. Qui plus est, la version 1.8 est exigée au minimum.

Téléchargez (voir figure 2-20) la version la plus récente d'Ant et procédez comme précédemment pour son installation. Pour toute autre plate-forme que Windows, ne choisissez pas l'archive au format `.zip`, car ce format ne conserve pas les droits des répertoires et fichiers qu'il abrite, et ceux-ci sont généralement cruciaux. Sous Windows, les archives au format .tar peuvent être extraites via un excellent utilitaire gratuit nommé 7-Zip (http://www.7-zip.org/). Rangez Ant dans un emplacement spécifique, modifiez les lignes habituelles dans le fichier `.bash_profile` par les suivantes, sauvegardez, appliquez les changements, vérifiez le bon fonctionnement du tout, c'est terminé.

```
ADT_SDK=~/dev/adt/sdk
export ANT_HOME=~/dev/ant
export JAVA_HOME=~/dev/jdk
export PATH=$PATH:$ADT_SDK/platform-tools:$ADT_SDK/tools:$JAVA_HOME/bin:$ANT_HOME/bin
```

Note

Certains des outils présentés dans ce chapitre tels que Node.js, le JDK ou encore Ant sont généralement disponibles via des gestionnaires de paquets (un peu comme npm) pour les utilisateurs avancés. Selon les systèmes d'exploitation, ceux-ci se nomment par exemple Chocolatey (https://chocolatey.org/), Npackd (https://npackd.appspot.com/), Homebrew (http://brew.sh/), MacPorts (http://www.macports.org/), APT (http://doc.ubuntu-fr.org/apt-get) et Yum (http://yum.baseurl.org/). Leur utilisation sort du cadre de cet ouvrage, mais il est toujours bon de connaître leur existence.

Figure 2-20
Téléchargement d'Ant sur son site officiel

Copyright © 2015 Eyrolles.

Android SDK Manager

La dernière étape dans l'installation du SDK est la gestion des différentes versions d'Android pour lesquelles vous souhaitez développer. Cela se passe dans ce qui se révèle être le point central du kit de développement : l'Android SDK Manager, un programme accessible via la commande `android`. Exécutez cette dernière pour démarrer l'utilitaire. L'interface de ce dernier assez simple (voir figure 2-21). Si tel n'est pas déjà le cas, décochez la case Obsolete et cochez Updates/New et Installed. Enfin, choisissez de classer les éléments de la liste par API level, il sera dès lors beaucoup plus simple de distinguer les nouveautés et mises à jour à installer des anciennes versions plus forcément très pertinentes, voire jugées désormais totalement obsolètes.

Figure 2-21
L'interface d'Android SDK Manager

Choisissez pour l'instant de ne télécharger que les versions les plus récentes des Android SDK Tools, Android SDK Platform-tools et Android SDK Build-tools sous la rubrique intitulée Tools ainsi que celles de la SDK Platform, et de la System Image correspondant à l'architecture de votre machine, sous la rubrique décrivant la version la plus récente d'Android (ici Android 4.4.2 – API 19). Vous pouvez également décider d'installer l'Android Support Library (aide à la compatibilité) et l'Intel x86 Emulator Accelerator sous la rubrique Extras. Pressez ensuite le bouton Install N packages… pour débuter le processus.

Une fenêtre, similaire à la figure 2-22, devrait s'ouvrir vous invitant à accepter chacune des licences indispensables à l'accès aux téléchargements. Après un certain délai dépendant de votre connexion à Internet, vous aurez à disposition tout le nécessaire pour permettre à Cordova/PhoneGap de compiler vos applications pour Android. Fermez ensuite l'Android SDK Manager.

Copyright © 2015 Eyrolles.

Figure 2-22
Accepter les licences avant d'accéder aux téléchargements

Command Line Interfaces

Avant d'installer les interfaces en ligne de commande Cordova et PhoneGap, il est important de signaler l'existence de trois modules Node.js dont l'installation est visiblement optionnelle, mais tout de même vivement conseillée. Les deux premiers d'entre eux s'adressent uniquement à iOS, et sont nommés respectivement ios-sim (https://github.com/phonegap/ios-sim) et ios-deploy (https://github.com/phonegap/ios-deploy). Comme leurs noms l'indiquent, ils servent à piloter le déploiement d'applications via le terminal vers l'émulateur ou bien un appareil iOS. Ces modules ne sont étrangement pas inclus au moment du téléchargement et de l'installation des CLI Cordova/PhoneGap et, bien qu'il soit possible d'utiliser Xcode pour arriver au même résultat, on préférera les ajouter pour nous abstraire du SDK :

```
sudo npm install --global ios-sim ios-deploy
```

Le dernier, plugman (https://github.com/apache/cordova-plugman/), permet une gestion complète des plug-ins – notamment dans le cadre d'une publication après création. Installez-le à l'aide de la commande suivante (encore une fois, hormis sous Windows, `sudo` peut être nécessaire) :

```
npm install --global plugman
```

Pour terminer, installons enfin, toujours via npm, les commandes `phonegap` (https://github.com/apache/cordova-cli) et `cordova` (https://github.com/phonegap/phonegap-cli) qui sont également des modules pour Node.js. Il n'est pas nécessaire d'utiliser les deux, cependant si vous débutez, essayez tour à tour l'un et l'autre pour déterminer avec lequel vous vous sentez le plus à l'aise :

```
npm install --global cordova phonegap
```

Patientez un court instant… Félicitations, vous venez juste de terminer l'installation de PhoneGap et Cordova sur votre ordinateur ! Voyons maintenant rapidement quelles peuvent être les similarités et différences de ces deux outils.

Copyright © 2015 Eyrolles.

Cordova CLI

L'interface en ligne de commande Cordova est relativement bas niveau. Un coup d'œil rapide à son manuel d'utilisation (cordova help, voir figure 2-23) confirme ceci de par la multitude de commandes et options bien spécifiques disponibles. Nous nous contenterons pour l'instant de décrire les plus importantes d'entre elles, à savoir :

- `cordova create` : à exécuter obligatoirement une fois, elle permet de démarrer un nouveau projet ;
- `cordova platform add` : également nécessaire, elle permet d'ajouter de nouvelles plates-formes pour lesquelles compiler l'application (à l'inverse, `cordova platform rm` retire des plates-formes précédemment ajoutées) ;
- `cordova plugin add` : bien qu'optionnelle mais généralement très utilisée, cette commande offre la possibilité d'ajouter des plug-ins afin d'enrichir le noyau `cordova` de nouvelles fonctionnalités (naturellement, `cordova plugin rm` est son opposée) ;
- `cordova run` : permettant le déploiement d'applications vers différents appareils et/ou simulateurs, elle sera probablement la commande dont vous vous servirez le plus ;`cordova serve` : elle démarre quant à elle un serveur web local facilitant et accélérant ainsi le développement sur n'importe quel navigateur de préférence pourvu d'outils de debug performants comme ceux de Google Chrome.

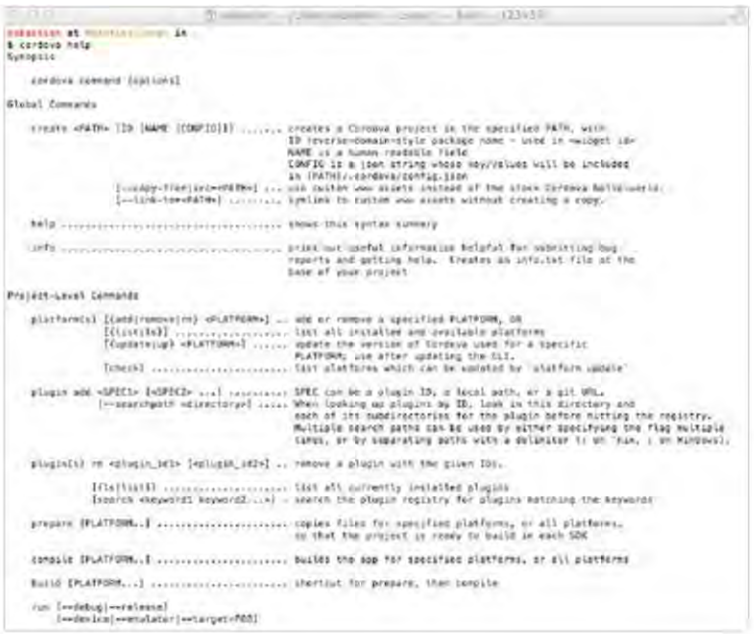


Figure 2-23
Un fragment de l'aide de Cordova CLI

Copyright © 2015 Eyrolles.

PhoneGap CLI

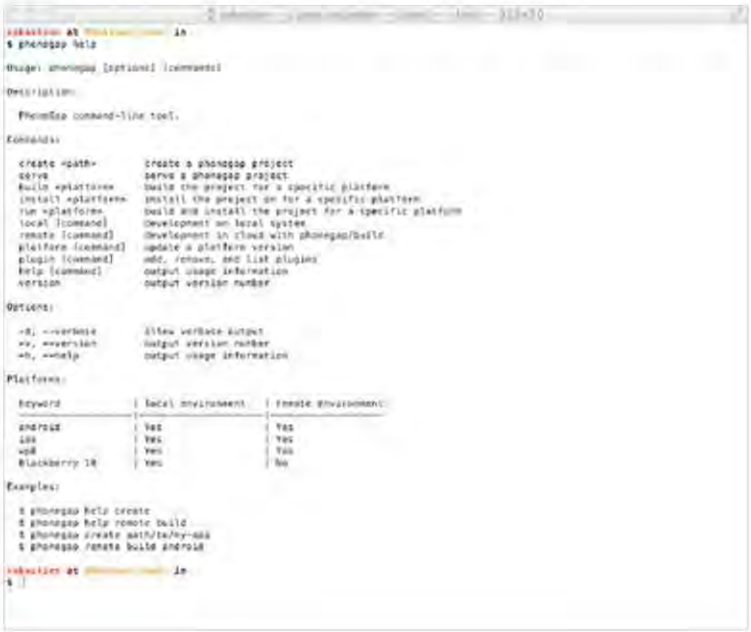
Basée sur Cordova CLI qu'elle utilise en arrière-plan, l'interface en ligne de commande PhoneGap est plutôt une sorte d'abstraction dont la principale vocation est d'apporter une gestion de PhoneGap Build dans le terminal. En effet, après une courte analyse de son manuel d'utilisation (phonegap help, voir figure 2-24), nous pouvons noter la présence de commandes équivalentes à celles présentées précédemment, avec parfois quelques légères différences :

- `phonegap create` ;
- `phonegap install` ;
- `phonegap plugin add/phonegap plugin remove` ;
- `phonegap run` ;
- `phonegap serve`.

Si leur portée est locale par défaut, certaines d'entre elles peuvent s'appliquer à PhoneGap Build, comme `phonegap install`. Dans ce cas, il s'agira d'ajouter le préfixe souhaité, au choix parmi `local` et `remote`, de la façon suivante : `phonegap local install ios` ou `phonegap remote install ios`. Enfin, `phonegap help` possède également la particularité d'afficher les plates-formes pour lesquelles le développement est possible en local selon le système d'exploitation utilisé (le développement iOS n'est, par exemple, possible que sous Mac OS X) ou bien à distance via PhoneGap Build. Plutôt pratique.

> **Note**
>
> Chacune des commandes composant Cordova et PhoneGap CLI doit obligatoirement être exécutée dans le répertoire de travail correspondant à celui du projet initié via `cordova create`/`phonegap create`, ou bien dans n'importe quel répertoire descendant de celui-ci.

Figure 2-24
L'aide de PhoneGap CLI



Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

Partie II

Développement d'une application

Votre âme de développeur doit à présent trépigner d'impatience à l'idée de commencer à produire du code ! Entrons donc dans le vif du sujet avec cette deuxième partie, qui va détailler toutes les étapes de la conception d'une application, depuis l'idée initiale jusqu'au debug sur un appareil physique, en passant par la gestion du code ou encore l'architecture des fichiers. Nous prendrons comme exemple une application réalisée spécifiquement pour cet ouvrage.

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

3

Création d'un projet

Réfléchir avant de se lancer

Tous les développeurs ont connu cette situation : vous découvrez une nouvelle technologie, que vous souhaitez utiliser pour réaliser un projet personnel et professionnel, et l'impatience se fait sentir ! Vous avez envie d'attraper le premier tutoriel trouvé et de développer le plus vite possible. Avec l'expérience, les développeurs apprennent qu'il faut résister à cette tentation et planifier son projet. Plus la préparation sera précise, plus le développement sera facile.

Choisir les plates-formes cibles

Avant de commencer le développement de votre application, il est important de se poser la question des plates-formes cibles. À l'heure où nous écrivons ces lignes, Cordova offre techniquement la possibilité de créer une application disponible sur Amazon Fire OS, Android, Bada, BlackBerry, FirefoxOS, iOS, Mac OS X, QT, Tizen, Ubuntu, WebOS et Windows Phone. Or, le choix des plates-formes cibles pour votre application est crucial pour plusieurs raisons.

Premièrement, votre application va être créée à destination d'un certain segment de marché (les jeunes, les plus de 50 ans, les enfants, les femmes, etc.). Chaque plate-forme dispose de sa propre audience, laquelle présente des comportements propres. Ainsi, si vous souhaitez proposer une fonctionnalité de paiement, sachez que les utilisateurs iOS sont généralement plus enclins à faire des achats in-app, contrairement aux utilisateurs Android.

Deuxièmement, chaque nouvelle plate-forme de destination amènera du travail en plus. Car même si la promesse de développement multi-plates-formes est globalement respectée, il restera toujours de petits détails à régler pour chaque OS, notamment en termes de design ou de correction de bugs.

Copyright © 2015 Eyrolles.

Ce qui nous mène au troisième point : les tests. Il est important que votre application soit consciencieusement éprouvée sur un ou plusieurs appareils de chaque plate-forme, de préférence sur un appareil physique et pas seulement dans un simulateur. Se procurer un iPhone 4, 5 ou 5s n'est pas forcément un défi, mais dénicher un appareil sous TizenOS ou Bada peut se révéler une véritable chasse au trésor.

Astuce

Les plates-formes Android et iOS sont saturées de certains types d'applications. Il peut être préférable d'en considérer d'autres moins convoitées : votre application pourrait plus facilement devenir une référence ! Il est en effet plus aisé d'être la meilleure application de Tetris sur Windows Phone que sur iOS…

Vous devez par conséquent connaître les quelques spécificités suivantes sur ces plates-formes.

- Les utilisateurs iOS sont plus enclins à effectuer des achats in-app et sont de grands consommateurs d'applications.
- Les utilisateurs iOS mettent très rapidement leur téléphone à jour lorsqu'une nouvelle version iOS est disponible, alors que les utilisateurs Android mettent peu à jour, surtout en raison des constructeurs qui ne mettent pas à disposition les nouvelles versions du système d'exploitation.
- Il existe de nombreuses résolutions d'écran chez Android, car le système est installé par différents constructeurs sur des appareils souvent très divers selon les gammes. À l'inverse, les appareils sous iOS possèdent pour le moment un nombre de résolutions plus facilement maîtrisable.
- Apple effectue une revue très stricte des applications, tandis qu'aucune revue n'est effectuée côté Google.
- Apple peut changer les règles de publication à tout instant et retirer une application de l'App Store sans préavis.
- Le téléchargement d'une application iOS ne peut se faire (légalement) que sur l'App Store, contrairement à Android qui autorise l'installation depuis des sources tierces.

Des étapes importantes avant le développement

Avant de débuter le développement de son application, il est bien évidemment primordial d'en définir le périmètre : le concept, la charte graphique, les fonctionnalités, etc. Il est de fait utile de suivre (avec plus ou moins de rigueur) les étapes qu'une agence professionnelle applique généralement :

- **définition du concept** (expliquez l'application en trois mots, trois phrases et trois paragraphes) **et de la cible** (qui est mon utilisateur cible ?) ;
- **spécifications fonctionnelles :** quelles sont les fonctionnalités de l'application ? (créer un compte, me connecter, poster une image, consulter les commentaires…) ;

Copyright © 2015 Eyrolles.

- **spécifications techniques :** quelles technologies vais-je utiliser (jQuery, AngularJS, Sencha Touch…) ? De quels plug-ins Cordova ai-je besoin ? Quelle solution de versioning de code vais-je utiliser ?
- **interface utilisateur :** quelles sont les pages de l'application, et de quelle manière s'enchaînent-elles ?
- **charte graphique :** est-ce que j'utilise un thème (gratuit ou payant) ? Est-ce que je crée un design de toutes pièces, à partir d'une liste d'applications qui m'inspirent ?

De nombreux outils ou ressources existent pour vous aider dans chacune de ces étapes. Pour réaliser le *wireframing* par exemple, c'est-à-dire le découpage de l'interface en différentes zones de l'interface utilisateur, vous disposez de Balsamiq (http://balsamiq.com/), Moqups (https://moqups.com/) ou UXPin (http://uxpin.com/).

Figure 3-1
Aperçu des possibilités offertes par Balsamiq

Astuce

L'étape de wireframing (sur papier ou dans un logiciel) peut paraître fastidieuse, mais permet de gagner beaucoup de temps lors du développement, et d'identifier dès le début les éléments qu'on pourra factoriser.

Concernant le choix d'une charte graphique, vous pouvez trouver de nombreux templates d'applications mobiles en téléchargement sur Internet, ou alors consulter des sites d'inspiration comme le Tumblr Mobile Design Inspiration (http://inspirationmobile.tumblr.com/) ou le blog Inspired UI (http://inspired-ui.com/).

Copyright © 2015 Eyrolles.

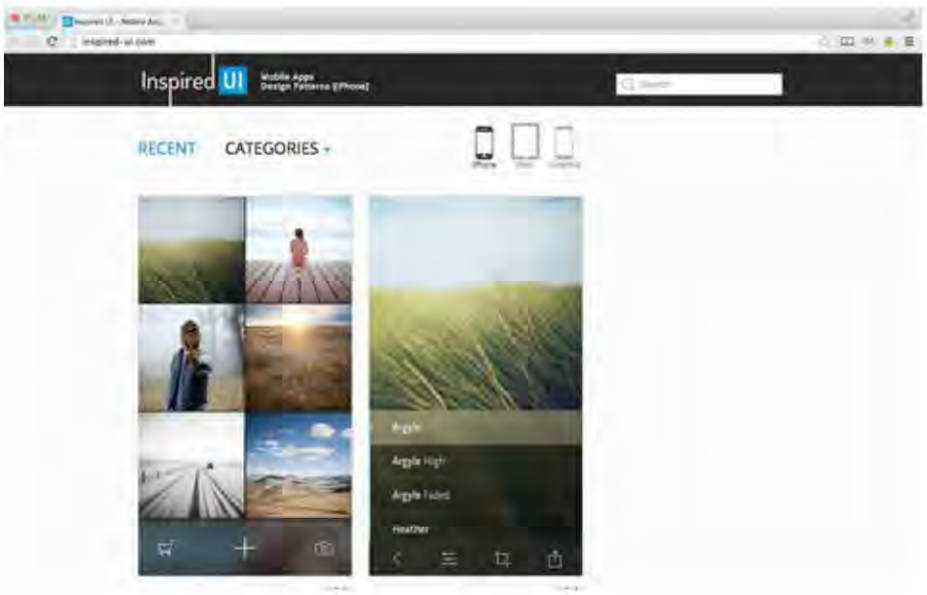


Figure 3-2
Page d'accueil du site Inspired UI

Les plug-ins Cordova

Pour chaque fonctionnalité définie dans les spécifications fonctionnelles, il faut cependant se poser les questions suivantes : puis-je réaliser cette fonctionnalité avec des technologies 100 % web ? Si non, un plug-in existe-t-il pour couvrir ce besoin ? Puis-je le développer moi-même (et bien sûr le partager sur GitHub !) ?

Pour information, à l'heure où nous écrivons ces lignes, voici la liste des plug-ins officiels Cordova :

- **Battery Status :** information sur l'état de la batterie ;
- **Camera :** gestion de l'appareil photo ;
- **Console :** surcharge du console.log JavaScript ;
- **Contacts :** gestion des contacts de l'appareil mobile ;
- **Device :** récupération d'informations sur l'appareil lui-même (système d'exploitation, nom, etc.) ;
- **Device Motion :** accès aux données de l'accéléromètre ;
- **Device Orientation :** accès à la boussole ;
- **Dialogs :** utilisation des fenêtres d'informations natives ;
- **File :** gestion de fichiers ;
- **File Transfer :** téléchargement et upload de fichiers ;
- **Geolocation :** accès aux données de localisation géographique ;
- **Globalization :** gestion de la langue et des fuseaux horaires de l'appareil mobile ;
- **In-App Browser :** affichage de sites web directement dans l'application ;

Copyright © 2015 Eyrolles.

- **Media :** lecture audio ;
- **Media Capture :** capture audio, vidéo et photo ;
- **Network Information :** accès aux informations réseau de l'appareil mobile ;
- **SplashScreen :** gestion de l'écran de chargement ;
- **Statusbar :** gestion d'état en haut de l'écran ;
- **Vibration :** gestion du vibreur.

Vérifiez bien leur compatibilité avec les plates-formes choisies en consultant la documentation associée.

Figure 3-3
Exemple d'un plug-in officiel hébergé sur GitHub

En plus de ces plug-ins officiels, GitHub regorge de nombreux autres plug-ins.

Astuce

Si vous trouvez un plug-in sur GitHub, mais qu'une fois mis en place vous constatez un bug, pensez à jeter un œil aux Pull Requests et Issues sur la page du projet GitHub (voir les détails dans les chapitres suivants). Cette section comprend toutes les propositions d'amélioration des développeurs.

Copyright © 2015 Eyrolles.

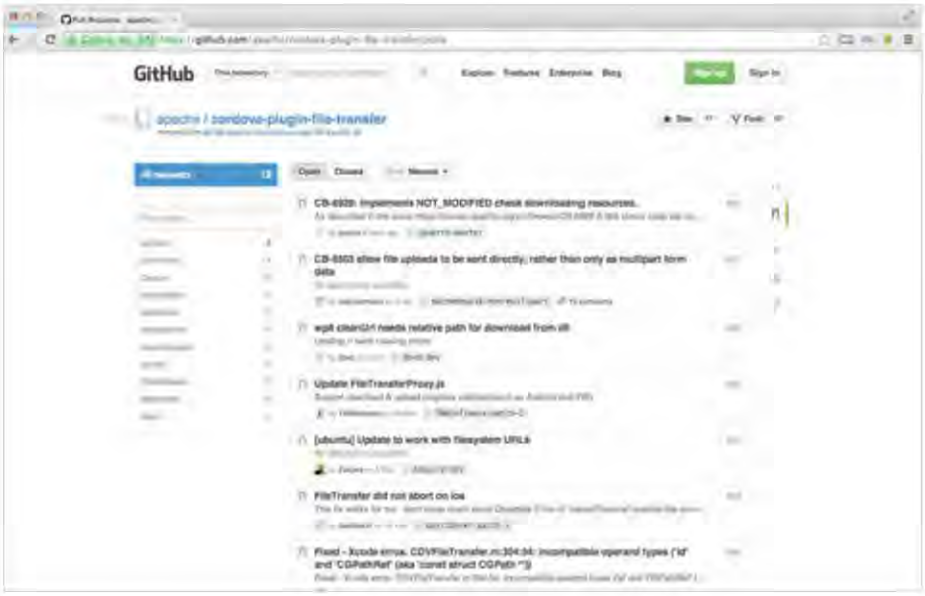


Figure 3-4
Liste des Pull Requests du plug-in File Transfer

Plug-in Cordova vs solution web

Il n'est pas toujours nécessaire de passer par un plug-in. Parfois une solution 100 % web existe. Pour illustrer cette situation, nous avons identifié deux exemples auxquels vous serez peut-être confronté.

Facebook Connect

Beaucoup d'applications, et peut-être la vôtre, permettent à l'utilisateur de se connecter en utilisant son compte Facebook. Les avantages sont multiples : l'utilisateur se connecte facilement, le développeur récupère des informations utiles, il est possible de préremplir le profil du visiteur, le partage sur Facebook se fait en un clic…

Cette fonctionnalité peut être implémentée de deux manières différentes : avec un plug-in Cordova ou de manière 100 % web.

Le plug-in Cordova dédié est disponible sur GitHub[1] et s'installe simplement comme tous les autres plug-ins. Il permet de tirer parti de l'application Facebook éventuellement déjà installée sur le téléphone. Autrement dit, au moment de la connexion dans votre application, l'utilisateur est redirigé vers l'application Facebook si elle existe. Il est alors automatiquement connecté. Il lui est simplement demandé d'autoriser votre application à accéder à son compte. Si Facebook n'est pas installé, une fenêtre s'ouvre dans votre application pour permettre à l'utilisateur de se connecter au site web et d'autoriser votre application.

1. https://github.com/phonegap/phonegap-facebook-plugin

Copyright © 2015 Eyrolles.

Christophe Coenraets, développeur connu de la communauté Cordova, a récemment mis en évidence sur son blog[2] qu'il était parfaitement possible de mettre en place ce même processus sans utiliser de plug-in. Il justifie sa solution en mettant en avant le fait que le plug-in Facebook officiel n'a pas toujours été très à jour des nouvelles versions du SDK Facebook et de Cordova. Il a donc créé une bibliothèque JavaScript qui tire profit du SDK Facebook JavaScript. Sa méthode fonctionne donc même pour les plates-formes non prises en charge par le plug-in Cordova. Mais d'un autre côté, l'intégration n'est pas aussi poussée, les fenêtres de connexion ne sont pas natives, et l'intégration avec l'application Facebook officielle n'est pas gérée.

Figure 3-5
Page du projet OpenFB de Christophe Coenraets sur GitHub

Prise de photos

La fonctionnalité de prise de photos dans une application met le développeur devant un choix similaire. En effet, il existe le plug-in officiel Cordova, mais également l'attribut `accept` sur la balise `input` :

```
<input type="file" accept="image/*;capture=camera">
```

Cette petite ligne d'HTML ajoute à l'application un bouton Choisissez un fichier, lequel déclenche l'ouverture de l'application photo sur un appareil mobile. Cet élément HTML s'affichera différemment suivant les navigateurs utilisés (Android, iOS…), mais il est tout de même possible de le personnaliser.

2. http://coenraets.org/blog/2014/04/facebook-phonegap-cordova-without-plugin/

Copyright © 2015 Eyrolles.

L'utilisation du plug-in permet toutefois un contrôle plus fin. On peut par exemple récupérer l'image au format base64, récupérer son chemin d'accès sur le système de fichiers en local, choisir le format et la qualité de l'image, ou encore de faire en sorte qu'une nouvelle photo prise soit sauvegardée dans l'album de l'appareil ou ailleurs.

Astuce

Un champ de type `file` n'a pas de style commun à toutes les plates-formes, et n'est pas vraiment adapté au mobile. Vous serez donc amené à le personnaliser. La manière la plus simple est d'appliquer une opacité nulle (`opacity: 0;`) en CSS sur l'élément, et de le placer dans un élément avec une image de fond représentant un beau bouton. L'utilisateur cliquant sur le bouton cliquera en réalité sur le champ invisible.

Quelle que soit la fonctionnalité, il est donc pertinent de se poser la question entre l'utilisation d'un plug-in ou un développement 100 % web. Si vous souhaitez également proposer une déclinaison WebApp de votre application, il fait sens d'opter pour la version web. Gardez cependant à l'esprit que le développement Cordova se passe en grande partie dans le navigateur, car cela est plus rapide et pratique, tandis que le test sur appareil mobile ne s'effectue que de temps en temps. Or les plug-ins ne fonctionneront que dans un simulateur ou sur un appareil de test et ne peuvent généralement pas être testés directement dans le navigateur. Ils peuvent même bloquer vos tests si vous avez par exemple besoin de commencer par soumettre une photo avant de poursuivre un processus dans l'application.

Astuce

Cordova évolue constamment et chaque mise à jour apporte son lot de nouveautés. Une des mises à jour récentes de Cordova comportait l'ajout de la plate-forme desktop, au même titre qu'iOS ou Android, permettant de proposer une version web des plug-ins. Ainsi, le plug-in de prise de photos dispose maintenant officiellement d'une version web utilisable pour vos WebApp. Tous les plug-ins n'ont pas encore proposé cette version, mais cela ne saurait tarder.

Ne pas réinventer la roue

En tant que développeur, vous serez souvent confronté à une situation où vous devrez choisir entre l'utilisation d'outils externes et le développement de vos propres briques. Nous aimerions vous donner quelques pistes pour que vous choisissiez la première option dans la plupart des cas.

Frameworks, bibliothèques et autres outils

Lorsqu'il est question de développement, que ce soit web, mobile ou autre, il est toujours important de ne pas réinventer la roue. Internet regorge de frameworks, bibliothèques et autres outils qui vous feront gagner du temps !

Copyright © 2015 Eyrolles.

Bibliothèques JavaScript

Une bibliothèque JavaScript est généralement composée d'un fichier JavaScript unique facilitant le développement ou apportant de nouvelles fonctionnalités. Voici les plus communes dans le domaine du développement hybride.

- **jQuery**, très connue, est une bibliothèque JavaScript à la fois puissante et légère, permettant de manipuler le DOM *(Document Object Model)*, c'est-à-dire de déplacer des éléments d'une page, de modifier leur style et leur contenu, etc. (http://jquery.com/)
- **lodash** est une bibliothèque d'utilitaires modulaire, extrêmement pratique, avec pour mots d'ordre performances et couverture de test. (https://lodash.com)
- **AngularJS** est un framework de plus en plus utilisé qui dispose d'une communauté très active. Il permet d'organiser/structurer son code en séparant la logique des vues, des données et des contrôleurs, en plus de fournir de nombreux mécanismes permettant de clarifier et d'améliorer le code en général. (https://angularjs.org/)
- **Sencha Touch** est un framework complet de développement JavaScript pour organiser son code en séparant la logique des vues des données et des contrôleurs, tout en fournissant également de nombreux widgets. (https://www.sencha.com/products/touch/)
- **Backbone** est une bibliothèque fournissant une manière d'organiser son code, en séparant les vues des données et des contrôleurs. (http://backbonejs.org/)
- **Fastclick** est une bibliothèque JavaScript très pratique pour augmenter la réactivité des interfaces mobiles en retirant le délai de 300 ms présent par défaut lorsqu'un utilisateur appuie sur un élément. (https://github.com/ftlabs/fastclick)
- **RequireJS** est une bibliothèque de chargement de fichiers, permettant d'optimiser le temps de chargement des applications en ne chargeant que les fichiers utiles au bon moment. Cette bibliothèque est principalement utilisée pour de la gestion de dépendances. (http://requirejs.org/)

Bibliothèques UI

Le terme UI fait référence à *User Interface*. Une bibliothèque UI améliore ainsi l'interface utilisateur.

- **jQuery Mobile** offre un large choix de widgets et un système de navigation entre les vues. Bien qu'assez décrié par une partie des développeurs pour sa complexité et sa lourdeur, il peut faire gagner du temps dans le cadre de développement d'une application de démonstration par exemple. (http://jquerymobile.com/)
- **Ionic**, basé sur AngularJS, est le framework UI proposant aujourd'hui les fonctionnalités les plus avancées, performantes et complètes pour mettre facilement au point une application mobile hybride de qualité et agréable à l'œil. (http://ionicframework.com)
- **Flat UI** est un ensemble de styles et de widgets pour créer rapidement un prototype d'application au design flat. (http://designmodo.github.io/Flat-UI/)
- **ChocolateChip-UI** est composé d'un ensemble de widgets (liste, pop-up, onglets…) et un système de navigation entre les vues. (http://chocolatechip-ui.com/)

Copyright © 2015 Eyrolles.

- **Font Awesome** est un ensemble de plus de 400 icônes disponibles sous forme d'une police de caractères. (http://fortawesome.github.io/Font-Awesome/icons/)
- **Glyphicon** est également une police de caractères contenant plus de 450 icônes. (http://glyphicons.com/)

Un choix important car structurant

Comme vous avez pu le voir, il existe de nombreuses bibliothèques, frameworks et outils à disposition. La liste présentée dans cet ouvrage n'est pas exhaustive, de nouveaux sont créés presque tous les jours. Chacun de ces outils dispose de ses fans et de ses détracteurs, et les débats sont légion sur ces sujets.

Même s'il est possible de développer une application sans aucun outil, il est tout de même conseillé de savoir bien s'entourer. Certains outils ont peu d'impact, comme jQuery qui peut être utilisé ponctuellement pour rendre un service et reste assez léger. En revanche, des outils comme AngularJS et Sencha Touch sont très structurants et très impactants, car ils vont définir la manière dont l'application va être entièrement développée. Et revenir sur un tel choix voudra souvent dire redévelopper une très grande partie de l'application.

Lors du choix d'un outil, considérez bien les points suivants :

- **la taille de la communauté :** des outils comme AngularJS bénéficient d'une très grande communauté de développeurs, qui seront à même de vous aider en cas de soucis ;
- **l'ancienneté :** un outil trop jeune représente un risque, surtout s'il est structurant, car son support peut s'arrêter du jour ou lendemain et la taille de sa communauté est probablement limitée ;
- **le nombre de contributions :** directement liée à la taille de la communauté, la quantité de contributions donne une indication sur les possibilités offertes par l'outil ;
- **la compatibilité :** si votre choix s'arrête sur plusieurs outils, vérifiez bien leur compatibilité. Sencha Touch ne fera pas bon ménage avec jQuery Mobile, par exemple, car les deux font beaucoup de choses en commun. De même, jQuery n'est peut-être pas utile en complément d'AngularJS qui offre déjà de larges possibilités ;
- **les performances :** un mauvais choix peut avoir un énorme impact sur les performances de votre application, tout comme un bon choix peut faire toute la différence ;
- **la couverture de tests :** un outil très bien couvert en termes de tests sera généralement de bonne qualité et plus robuste dans la durée ;
- **la licence de distribution[3] :** l'outil peut être distribué de manière gratuite ou payante, et sa licence peut parfois vous empêcher de vendre votre application !

3. http://choosealicense.com/

Copyright © 2015 Eyrolles.

Architecture et structure

Un projet Cordova ne se construit pas n'importe comment. Son organisation doit respecter des règles strictes que nous allons passer en revue.

Architecture d'un projet Cordova par défaut

Comme nous l'avons vu précédemment, la commande `cordova create` crée un nouveau projet Cordova directement en ligne de commande. Après son exécution, vous pourrez constater, dans le dossier de destination, l'architecture de base de tout nouveau projet Cordova.

- `www` : ce dossier contient l'ensemble des fichiers web nécessaires au fonctionnement de votre application (HTML, JS, CSS, images, etc.) Lors de la création du projet, ce dossier accueille une application de démonstration, permettant de tester facilement le fonctionnement de `cordova`.
- `plugins` : ce dossier contient les fichiers sources des plug-ins installés via la commande `cordova plugin add`.
- `platforms` : ce dossier contient le code complet du projet d'application (pour chaque plate-forme ajoutée via la commande `cordova platform add`), organisé par plates-formes (`ios`, `android`, `wp7`...). Le processus de compilation Cordova, initié par la commande `cordova build ios` par exemple se charge de déplacer le contenu des dossiers `www` et `plugins` dans le dossier de la plate-forme pour laquelle vous compilez.
- `merges` stocke des fichiers spécifiques à chaque plate-forme, comme une feuille de styles différente. Si votre dossier `www` embarque un fichier `style.css`, votre dossier merges peut par exemple contenir un fichier `ios/style.css` et/ou `android/style.css` qui remplacera automatiquement le fichier initial lors de la compilation.

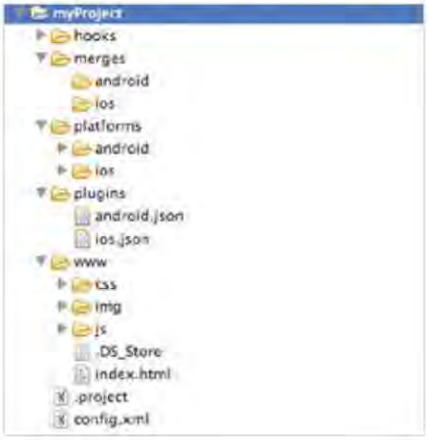


Figure 3-6
Structure de base d'une application Cordova, affichée dans Eclipse

Fichier de configuration

Au même niveau que le dossier `www` d'un projet Cordova doit obligatoirement se trouver un fichier de configuration nommé `config.xml`. Celui-ci contient de nombreuses informations permettant de personnaliser le fonctionnement de votre application.

Copyright © 2015 Eyrolles.

Pour rappel, lors de la création d'un projet Cordova, le dossier `www` accueille une application de démonstration, avec, notamment, un fichier `config.xml` basique dont voici le contenu détaillé :

- précision du type d'encodage du fichier :

```
<?xml version="1.0" encoding="utf-8"?>
```

- informations générales sur le projet, et notamment l'identifiant de l'application au format dit « nom de domaine inversé » (reverse domain style) :

```
<widget id="io.cordova.hellocordova" version="0.0.1" xmlns="http://www.w3.org/ns/
widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0">
```

- nom de l'application, c'est celui qui apparaît sous l'icône sur l'écran d'accueil d'un appareil mobile :

```
    <name>HelloCordova</name>
```

- description de l'application :

```
    <description>
        A sample Apache Cordova application that responds to the deviceready event.
    </description>
```

- identité de l'auteur de l'application :

```
    <author email="dev@callback.apache.org" href="http://cordova.io">
        Apache Cordova Team
    </author>
```

- adresse de la première page de l'application par rapport à la racine du dossier `www`. Si la page d'accueil de votre application est située sous `www/app/main.html`, vous pouvez le préciser ici :

```
    <content src="index.html" />
```

- précision des adresses menant à des ressources externes dont l'accès est autorisé depuis l'application, via un webservice par exemple. Vous pouvez restreindre ce paramètre pour des raisons de sécurité en précisant un domaine particulier (par exemple : `"http://*.google.com"`, `"http://google.com"`, `"https://google.com"`…) :

```
    <access origin="*" />
```

- précision du mode d'affichage de l'application, en plein écran (masque la barre d'état) ou non :

```
    <preference name="fullscreen" value="true" />
```

- par défaut, l'utilisateur pourra scroller l'application au-delà de l'écran en haut et en bas, provoquant un « rebond » de l'application :

```
    <preference name="webviewbounce" value="true" />
```

- balise fermante du widget décrivant l'application :

```
</widget>
```

Copyright © 2015 Eyrolles.

De nombreuses autres configurations sont accessibles pour ce fichier, que ce soit des configurations générales ou propres à une plate-forme donnée. Référez-vous à la documentation officielle pour plus de détails.

Ce fichier peut être édité comme un fichier texte, ou visualisé et modifié en tant que fichier XML par un logiciel spécialisé.

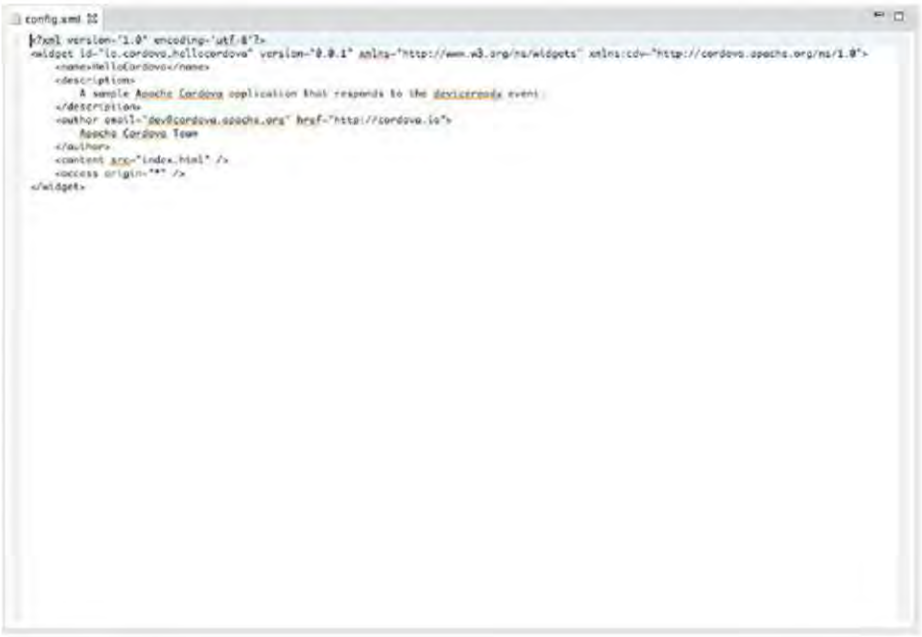


Figure 3-7
Fichier de configuration édité au format texte

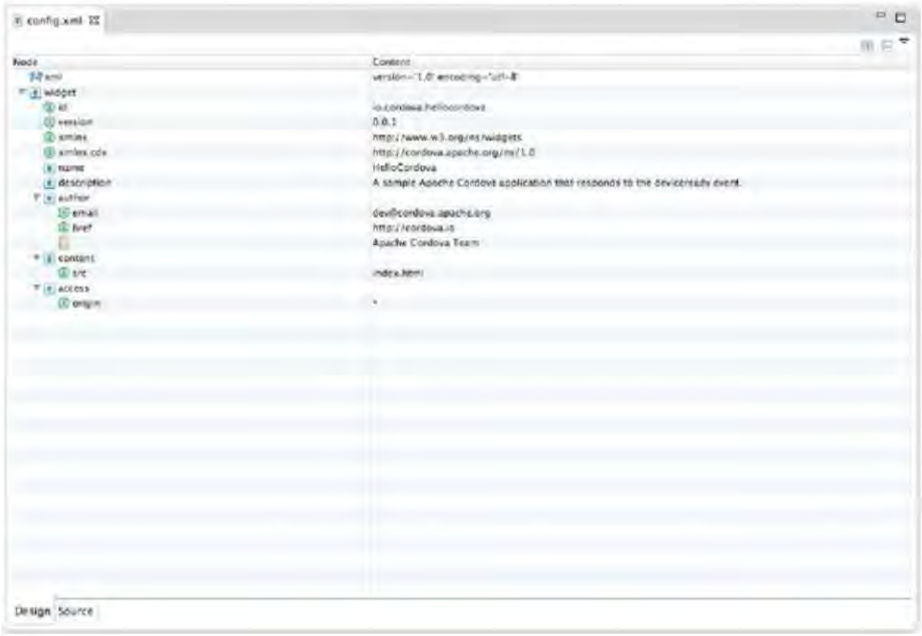


Figure 3-8
Fichier de configuration édité au format XML

Copyright © 2015 Eyrolles.

Application de démonstration

La commande `cordova create` sert à créer un projet par défaut contenant une application de démonstration dont les sources ont été placées dans le dossier `www`. Le but de cette application est simplement de tester le fonctionnement de Cordova dans vos divers simulateurs ou sur vos divers appareils.

Ouvrez par exemple le fichier `index.html` du dossier `www` avec votre navigateur Internet préféré. Vous constaterez que l'application indique CONNECTING TO DEVICE. Cette instruction indique que la « connexion » avec Cordova ne se fait pas, ce qui est logique, car on affiche l'application dans un navigateur web en tant que page web, et non en tant qu'application mobile sur un appareil mobile.

Figure 3-9
Application de démonstration dans le navigateur Chrome

Si vous déployez l'application dans un Simulateur iPhone par exemple (nous verrons par la suite comment procéder), vous constaterez que la « connexion » avec Cordova s'effectue correctement et que la mention DEVICE IS READY s'affiche.

Figure 3-10
Application de démonstration dans le simulateur iPhone

Astuce

La majorité de votre application pourra probablement être testée dans Chrome, sans que la « connexion » avec Cordova ne soit effective. Mais pensez à tester régulièrement dans un simulateur ou sur un appareil mobile !

Copyright © 2015 Eyrolles.

Utiliser un squelette personnalisé

Cette application de démonstration est une bonne manière de vérifier que tout fonctionne bien avant de commencer à développer en partant ainsi d'une base saine. Mais cette dernière peut vite se révéler insuffisante et vous ressentirez peut-être l'envie de la personnaliser.

La commande `cordova create src=<PATH>` permet de spécifier un contenu alternatif pour le dossier `www`. Si vous développez de nombreuses applications, vous pouvez ainsi créer un modèle d'application, contenant un squelette de base qui importe déjà l'ensemble des frameworks avec lesquels vous travaillez, comme jQuery par exemple. Vous pouvez également inclure dans le dossier `www` vos icônes et écrans de lancement (splashscreen) par défaut pour chaque nouvelle application en cours de développement.

Note

Si vous ne précisez pas le nom du projet dans la commande `cordova create src=`, c'est le nom du dossier précisé dans l'attribut `src` qui sera utilisé comme nom de projet

Versionner ses sources

Il ne vous viendrait pas à l'esprit de conserver vos photos de vacances sans en faire une ou plusieurs sauvegardes ? De même, il ne serait pas envisageable de vous passer du raccourci Ctrl + Z pour annuler une modification dans un document Word ? Saviez-vous que des outils vous permettent de sauvegarder votre code, mais aussi d'effectuer un Ctrl + Z à tout moment, sur l'ensemble de votre code, depuis la première ligne écrite ?

Pourquoi faire du versioning ?

Pour un développeur averti, versionner ses sources semble être une évidence, mais il est important de rappeler la définition et l'intérêt de cette pratique. Le *versioning* de code est une technique qui stocke l'ensemble des fichiers de votre application, tout en conservant la chronologie des modifications qu'ils ont subies. Cela offre plusieurs avantages.

Le premier est de sauvegarder l'historique des modifications d'un fichier. Il est ainsi plus facile de revenir en arrière après une erreur malheureuse.

Mais le principal intérêt du versioning se révèle lors du travail en équipe. Travailler à plusieurs sur la même base de code en simultané devient très compliqué si on ne dispose pas de tels outils. Par exemple : Pierre et Marc sont en train de travailler sur le fichier `index.html`. Pierre sauvegarde son travail, suivi de Marc. S'ils travaillent sur Dropbox, deux fichiers en conflit vont être créés, il faudra alors les fusionner à la main. S'ils travaillent sur un serveur FTP, c'est la catastrophe, car la version de Pierre aura été totalement écrasée. Et pour peu que Pierre et Marc ne soient pas en contact permanent et qu'aucune erreur n'apparaisse dans le fonctionnement de l'application, cet écrasement passera inaperçu.

Les outils de versioning en revanche vont détecter ce conflit, et dans la plupart des cas le résoudre d'eux-mêmes en fusionnant les modifications. Dans des cas plus complexes, l'outil présentera à Pierre (le dernier à sauvegarder) les différences entre son fichier et celui de Marc, pour qu'il règle le conflit à la main.

Copyright © 2015 Eyrolles.

Enfin, les outils de versioning présentent un autre avantage : pouvoir travailler sur une version alternative du code, sans modifier la version « en production ». Vous pouvez ainsi travailler à la correction d'un bug sans modifier le code principal, pour ensuite fusionner sa correction avec le code principal.

Outils de versioning

Il existe de nombreux outils de versioning, mais les trois principalement utilisés par les développeurs sont Git, SVN et Mercurial.

SVN propose une approche centralisée, là où GIT et Mercurial offrent une approche décentralisée.

Un système centralisé fonctionne avec une approche client/serveur : tous les fichiers sont stockés sur un serveur auquel les clients envoient leur code.

Le système décentralisé fonctionne avec une base commune de code sur le serveur, mais une base alternative sur chaque client. Ces derniers peuvent ainsi sauvegarder leurs modifications en local, puis les envoyer sur le serveur. Dans un système décentralisé, il est possible de *commiter* son code (autrement dit de le sauvegarder) en local sur sa machine, même si on n'est pas en mesure de contacter le serveur principal. L'intérêt est donc que chacun possède une copie au cas où.

SVN, également appelé Subversion, a pendant longtemps été l'outil le plus utilisé. Mais depuis quelques années, Git est utilisé par la majorité des développeurs, en raison de son approche décentralisée, mais aussi grâce à la plate-forme GitHub qui permet aux développeurs de partager leur code avec la communauté.

Dans le cadre de cet ouvrage, nous nous limiterons à l'utilisation de Git.

Fonctionnement de Git

Cet ouvrage n'a pas la prétention d'apprendre toutes les subtilités de Git, mais nous allons faire le tour des fonctionnalités principales. Nous simplifierons ainsi volontairement le fonctionnement de Git, qui peut s'avérer très complexe lorsqu'on entre dans les détails.

Git se base sur un *remote repository,* autrement dit un serveur de données distant, et une multitude de *local repositories,* autrement dit de multiples serveurs de données locaux, un sur chaque machine de chaque développeur participant au projet. Un fichier peut ainsi exister en trois exemplaires : une version en cours d'édition en local, une version sauvegardée sur le dépôt local du développeur et une version sauvegardée sur le dépôt distant. Chaque repository (*remote* ou *local*) stocke un historique de l'ensemble des versions de chaque fichier.

Voici un exemple de scénario de travail avec Git.

1. Le développeur commence son travail en récupérant l'ensemble des dernières modifications sur le remote repository (il fait un `pull`). Si le développeur n'a jamais travaillé sur ce projet auparavant, il récupère l'ensemble des sources (il fait un `checkout`).
2. Le développeur travaille sur l'ajout d'une fonctionnalité à son application. Il va ainsi modifier plusieurs fichiers pour ajouter la fonctionnalité à son application en local. Il sauvegarde alors ses modifications sur le local repository (il fait un `commit`). Pour le moment, les autres développeurs ne voient pas le résultat de son travail.

Copyright © 2015 Eyrolles.

3. Le développeur va travailler sur une seconde fonctionnalité, et ainsi de suite, en commitant (sauvegardant) son code en local à chaque étape.
4. Lorsque le développeur veut partager son travail avec le reste de l'équipe, il va récupérer les sources actuelles (`pull`). Cette étape est importante, car pendant son temps de travail, le reste de l'équipe a probablement également développé, et ses sources en local ne sont plus à jour.
5. Si ses coéquipiers ont travaillé sur les mêmes fichiers que lui, Git va automatiquement fusionner leurs modifications avec les siennes. Dans certains cas particuliers, s'il ne parvient pas à réaliser la fusion tout seul, il va modifier le contenu du fichier en question en indiquant les sections du développeur en conflit avec celles de son équipe. Il faudra alors résoudre le conflit à la main.
6. Lorsque les sources ont été récupérées et tous les fichiers fusionnés avec succès, le développeur peut envoyer son code sur le serveur pour le mettre à disposition des autres (il fait un `push`).

Astuce

Pensez toujours à bien faire un `pull` des sources avant d'effectuer un `push` de votre propre code.

Git : commandes de base

Voici quelques commandes de base pour gérer un système de fichier avec Git, à exécuter dans une fenêtre de terminal.

```
git init
```

Crée un repository à l'endroit où la commande est exécutée.

```
git clone username@host:/path/to/repository
```

Récupère le contenu d'un repository en ligne.

```
git add <filename>
```

Ajoute un fichier au système Git. Doit être effectué pour chaque nouveau fichier créé.

```
git commit -m "Commit message"
```

`commit` en local l'ensemble des modifications, accompagné d'un message explicatif.

```
git push origin master
```

Envoie l'ensemble des commits locaux sur le remote repository.

```
git pull
```

Récupère la dernière version de chaque fichier sur le remote repository.

```
git checkout -- <filename>
```

Annule les modifications sur un fichier local et récupère la version sur le remote repository.

Copyright © 2015 Eyrolles.

```
git fetch origin
git reset --hard origin/master
```

Annule toutes les modifications locales et revient à l'état du remote repository.

Astuce

Vous pouvez utiliser un gestionnaire de projet comme Eclipse avec le plug-in EGit ou encore un logiciel tel que Tower pour Mac (http://git-tower.com), permettant d'avoir accès à ces commandes via une interface graphique.

Utiliser GitHub

Git permet à des équipes de travailler facilement sur un projet en commun, en simultané et sans risque de conflits. Vous pouvez ainsi développer votre application en collaboration avec les développeurs que vous connaissez. Mais vous aimeriez peut-être recevoir l'aide d'autres développeurs autour du monde, pour vous aider à améliorer la sécurité, simplifier le code, trouver un bug ou développer de nouvelles fonctionnalités ? GitHub vous offre cette possibilité.

En effet, il s'agit d'un annuaire de projets Git, qui héberge le code d'un projet, mais aussi permet d'en faire la présentation. Les développeurs du monde entier peuvent donc présenter leurs réalisations, mettre à disposition leur code et éventuellement recevoir l'aide d'autres développeurs pour l'améliorer.

Figure 3-11
Exemple d'un projet GitHub hébergeant le code d'une application Cordova personnelle

Copyright © 2015 Eyrolles.

Astuce

GitHub peut représenter votre « CV de développeur », permettant aux employeurs potentiels de regarder votre code et évaluer vos compétences. Soignez donc la présentation de vos projets, ne négligez pas les commentaires dans le code, et essayez de collaborer sur d'autres projets pour montrer votre dynamisme !

L'inscription à GitHub est très simple. Rendez-vous simplement sur http://github.com et suivez les instructions pour créer votre premier repository. La présentation du projet s'effectue dans un fichier `README.md`. Le nom doit être strictement identique et le contenu écrit au format Markdown, qui est un langage d'écriture permettant de mettre en forme du texte de manière très simple. Il existe un cours en ligne sur ce sujet[4].

Pour chaque projet, vous pouvez accéder à la liste des contributeurs du projet et la liste des commits (c'est-à-dire les modifications apportées à chaque fichier). Un bouton Download ZIP permet également de télécharger le code source sans utiliser la commande `git clone`.

Figure 3-12
Historique des commits d'un projet sur GitHub

4. http://fr.openclassrooms.com/informatique/cours/redigez-en-markdown

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

4

Conception et architecture d'une application

Pour illustrer les grands principes du développement avec Cordova, nous allons passer en revue les étapes de conception d'une application pour iOs et Android, réalisée spécifiquement dans le cadre de ce livre. Puis nous décrirons l'architecture de cette application.

Étapes de conception

Il nous a semblé primordial de prendre un exemple concret qui rassemble tous les concepts propres à Cordova, tout en restant une application mobile pertinente. Notre choix s'est arrêté sur une application de journal de bord, baptisée « Rappelle-toi ».

Le fonctionnement de l'application est très simple puisqu'elle s'articule autour d'une *timeline* contenant les différents « articles » publiés dans un journal, chacun correspondant à une année précise et à un type de données spécifiques. Nous avons choisi de proposer les types suivants : photo, vidéo, géolocalisation et texte simple. L'application sera développée pour iOS et Android, sans aucun framework structurant de programmation.

Les contours de l'application étant à présent posés, intéressons-nous aux différentes étapes du développement. Quelle que soit votre application, nous vous invitons à les suivre.

Étape 1 : définir le concept

L'application est un journal de bord servant à ajouter de nouveaux articles par année donnée. Un article peut contenir des photos, des vidéos, la géolocalisation ou un texte simple. Le concept peut bien sûr évoluer tout au long du développement, mais sa définition claire et simple permettra par la suite de prévoir les fonctionnalités – ce qui influencera le choix des plug-ins par exemple.

Copyright © 2015 Eyrolles.

Étape 2 : choisir les plates-formes cibles

Nous souhaitons que cette application soit disponible à la plus grande part de marché possible. Le choix d'iOS et d'Android est donc pertinent puisqu'ils couvrent à eux deux la majorité du marché des smartphones. L'ajout de Windows Phone ou BlackBerry demanderait du travail supplémentaire d'adaptation et de tests spécifiques.

Étape 3 : créer les wireframes

Afin de nous mettre d'accord, mais aussi de gagner du temps sur l'étape suivante, nous avons réalisé les wireframes sur Balsamiq. Cette étape est la première du processus permettant de se rendre compte des différences possibles de vision entre les développeurs, ou entre vous et vos futurs utilisateurs, si vous décidez de les inclure dans la discussion ! En effet, il est tout à fait possible d'échanger avec de futurs utilisateurs dès le début de la conception pour être sûr de répondre au mieux à leurs attentes. Il s'agit d'une étape où les différentes « écoles » du mobile peuvent s'affronter. Est-ce qu'on opte pour un menu qui s'ouvre sur le côté ou pour des onglets ? Est-ce que les messages d'information ou d'alertes sont des pop-ups natives ou des intersticiels ? Est-ce que l'indicateur de chargement prend la forme d'un petit GIF animé à côté de l'élément qui a été activé ou d'un écran bloquant sur toute la page ? Quelles transitions de pages utiliser ? Est-ce qu'Android et iOS disposent d'un design différent ?

Dans le cas de notre application, nous avons opté pour certains choix précis…

- La navigation se fera via un menu qui apparaît en haut de l'application, sous forme d'onglets, en raison du faible nombre d'entrées.
- Il n'y aura pas de différence de style entre iOS et Android.
- Le chargement prendra la forme d'un petit GIF animé qui apparaîtra à côté de l'intitulé du bouton qui vient d'être cliqué, avant de disparaître une fois le chargement terminé et la nouvelle page s'affichant.

N'hésitez pas à utiliser des outils comme inVision pour échanger sur vos wireframes, soit au sein de l'équipe de développeurs, soit avec votre direction ou vos clients, ou encore avec vos utilisateurs. Mais n'oubliez pas : le but est de développer une application, pas de gagner un concours de wireframes. Ne passez donc pas trop de temps sur cette étape qui doit rester aussi abstraite que possible.

Figure 4-1
Le service Invision

Copyright © 2015 Eyrolles.

Étape 4 : identifier les fonctionnalités

La création des wireframes permet d'amorcer une discussion sur les fonctionnalités, comme le type de connexion utilisé (Facebook Connect ? nom d'utilisateur/mot de passe ?). Attention toutefois : toutes les fonctionnalités ne sont pas visibles sur les wireframes, cette étape est donc primordiale. Par exemple, les wireframes ne montrent pas si une connexion automatique est mise en place, ou encore l'endroit où seront stockées les données affichées/collectées dans l'application. Le découpage en fonctionnalités vous permettra de prioriser leur développement et de les assigner aux différents membres de l'équipe. Les fonctionnalités peuvent s'exprimer sous différentes formes. Quelques exemples avec la connexion :

- « Facebook Connect » ;
- « Permettre la connexion via Facebook » ;
- « Je peux me connecter avec mon compte Facebook ».

Pour plus de clarté, nous vous conseillons la forme « je peux… » qui est centrée sur l'utilisateur et ses actions. Ne vous formalisez cependant pas trop, vous verrez ci-après que cette formulation n'est pas adaptée à toutes les fonctionnalités. Sans chercher à devenir un gourou de la gestion de projets, choisissez simplement une formulation avec laquelle vous êtes à l'aise.

Dans le cas de notre application, la liste est la suivante :

- je peux voir la timeline de mes articles, par ordre chronologique ;
- je peux afficher ou masquer le menu en haut de l'application ;
- je peux ajouter un article de type Photo ;
- je peux ajouter un article de type Vidéo ;
- je peux ajouter un article de type Géolocalisation ;
- je peux ajouter un article de type Texte Simple ;
- mes articles sont stockés dans le Local Storage[1] dans l'application.

Nous avons volontairement gardé cette liste de fonctionnalités la plus simple possible. Pour information, voici quelques questions à se poser pour définir des fonctionnalités avancées et les réponses que nous y avons apportées.

- Que se passe-t-il si je perds la connexion Internet pendant l'utilisation de l'application ? Dans notre cas, la connexion Internet n'est pas utilisée, il n'y a donc aucune incidence.
- Que se passe-t-il si je n'ai pas de connexion Internet au lancement de l'application ? Dans notre cas, la connexion Internet n'est pas utilisée, il n'y a donc aucune incidence.
- L'application fonctionnera-t-elle si je l'ouvre en mode paysage sur une tablette ? L'application s'affiche en mode portrait.
- Que se passe-t-il si j'atteins la limite de stockage possible dans le Local Storage ?
- Que se passe-t-il si j'ajoute un article à une année étant déjà renseignée ? L'entrée est remplacée.

1. http://diveintohtml5.info/storage.html

Copyright © 2015 Eyrolles.

- Quelle est la taille limite de texte que je permets lors de l'ajout d'un article de type Texte Simple ? Nous avons décidé de fixer la limite à 1 000 caractères, en bloquant la taille maximale de l'élément `textarea`.
- Concernant l'ajout de photos, est-ce que je permets l'ajout depuis la galerie de l'appareil, ou la prise de photos uniquement, ou les deux ? Nous avons décidé de permettre les deux cas d'usage.

Tout au long du développement, de nombreuses questions de ce type vont émerger. Nous vous faisons confiance pour les aborder de la meilleure des manières !

Étape 5 : identifier les plug-ins

Une application Cordova se compose généralement d'un ou plusieurs plug-ins. Il est important d'en faire la liste en se basant sur les fonctionnalités, et de vérifier que chacun d'eux est disponible pour les plates-formes ciblées.

Dans notre cas, nous utiliserons les plug-ins suivants :

- Camera : pour permettre la prise de photos ;
- Media Capture : pour permettre la prise de vidéos ;
- Geolocation : pour accéder à la position géographique de l'appareil ;
- SplashScreen : pour gérer plus finement l'affichage de l'écran de chargement.

Nous nous sommes assurés que tous ces plug-ins étaient bien disponibles pour iOS et Android et compatibles avec la dernière version de Cordova.

Étape 6 : créer le repository

Comme nous l'avons vu précédemment, il est primordial de versionner son code, pour pouvoir facilement revenir en arrière ou encore collaborer à plusieurs sur une même base de code.

La procédure de création d'un repository est relativement simple.

Créer un compte

Connectez-vous ou créez-vous un compte sur le site http://github.com.

Figure 4-2
Page d'accueil de GitHub

Copyright © 2015 Eyrolles.

Créer le repository

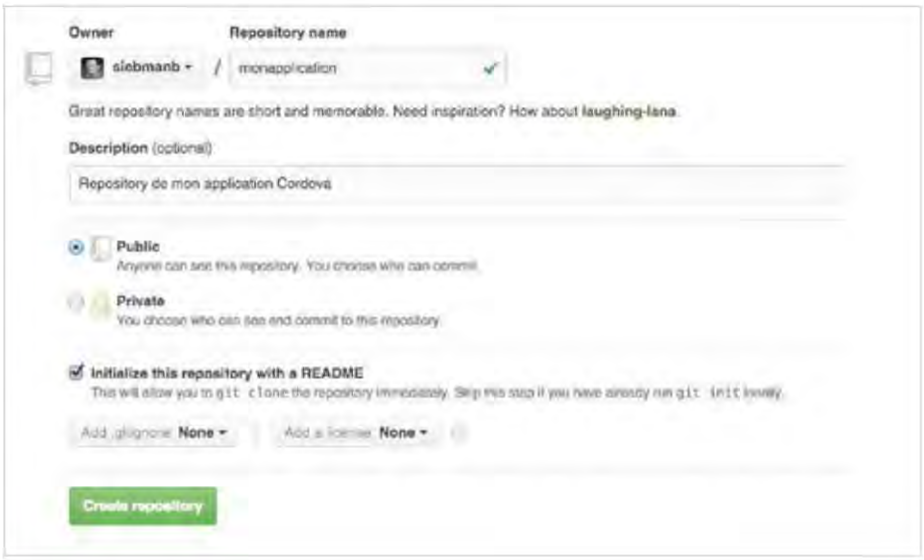
Dans la barre de menus, cliquez sur + puis sur New repository.

Figure 4-3
Ajout d'un repository

Configurer le repository

Choisissez un nom de repository et une description. Si vous désirez que votre code reste privé, vous pouvez choisir l'option Private, mais celle-ci requiert un compte GitHub payant. Ne choisissez pas d'initialiser le repository avec un fichier `README`, nous le ferons dans un second temps.

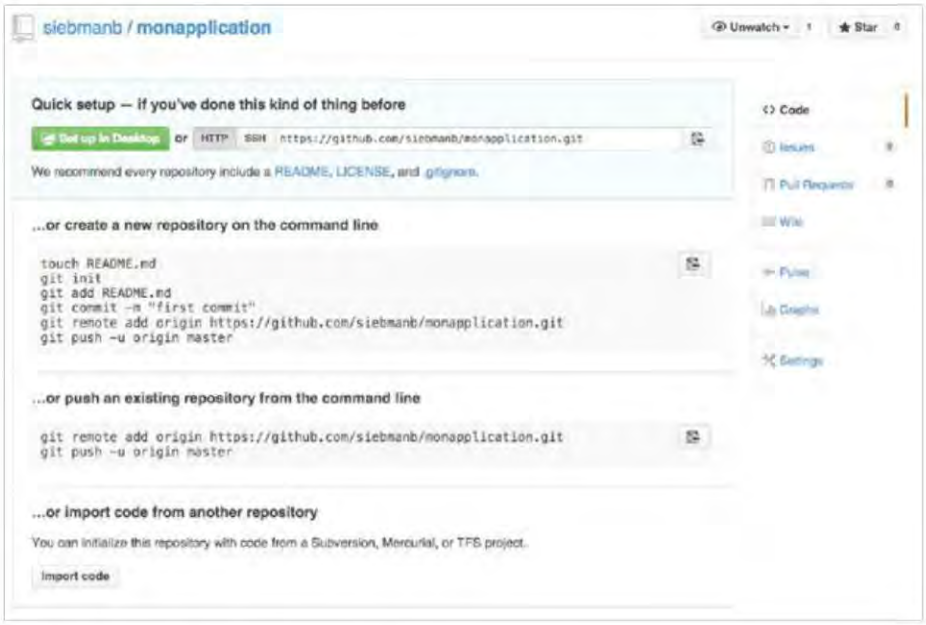
Figure 4-4
Configuration du repository



Votre repository GitHub est prêt ! Vous devez maintenant le synchroniser avec votre code local.

Copyright © 2015 Eyrolles.

Figure 4-5
Import du repository

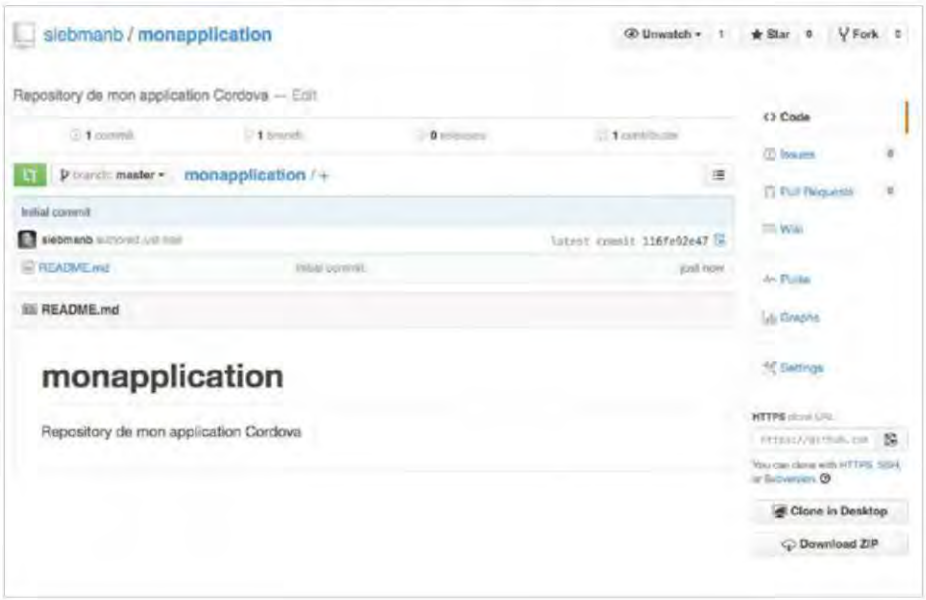


Importer le repository sur votre machine

Suivez simplement les instructions de GitHub dans la rubrique correspondant à votre situation :

- « …or create a new repository on the command line » : si vous n'avez pas encore réalisé de développement, ou la rubrique ;
- « …or push an existing repository from the command line » : si vous avez déjà produit du code.

Figure 4-6
Aperçu du repository



Copyright © 2015 Eyrolles.

Étape 7 : créer le projet Cordova

Créer le projet Cordova

Utilisez maintenant la commande.

```
cordova create monapp
```

Elle crée le dossier `monapp`, contenant l'architecture de base d'une application de démonstration.

Ajouter les plates-formes de destination

Ajoutez maintenant les plates-formes de destination avec lesquelles votre application sera compatible. Dans notre cas, iOS et Android.

```
cordova platform add ios
cordova platform add android
```

Ajouter les plug-ins

Ajoutez les plug-ins qui vous seront utiles. Voici les commandes d'ajout dans le cadre de notre application :

```
cordova plugin add org.apache.cordova.camera
cordova plugin add org.apache.cordova.media-capture
cordova plugin add org.apache.cordova.geolocation
cordova plugin add org.apache.cordova.splashscreen
```

Personnaliser le fichier config.xml

Le contenu du fichier de configuration de notre application est décrit en détail dans la suite du chapitre. N'hésitez pas à consulter la documentation officielle pour voir toutes les possibilités de configuration.

Déployer dans un simulateur

Comme nous l'avons vu précédemment, le déploiement sur un simulateur permet de tester votre application dans des conditions quasi réelles. La commande suivante compile l'application sous forme d'un projet Xcode.

```
cordova build ios
```

Ensuite, rendez-vous dans le dossier `platforms/ios` de votre projet et double-cliquez sur le fichier `*.xcodeproj` pour lancer Xcode. Nous expliquons dans le chapitre 6 comment déployer l'application dans le simulateur de votre choix. Vous pouvez également utiliser la commande suivante pour ouvrir directement un simulateur :

```
cordova emulate ios
```

Pensez à employer cette commande chaque fois que vous souhaitez voir vos dernières modifications appliquées.

Copyright © 2015 Eyrolles.

La commande suivante compile l'application sous forme d'un projet Android.

```
cordova build android
```

Puis vous pourrez déployer l'application dans un simulateur GenyMotion (voir chapitre 6).

Déployer sur un appareil

Si vous avez suivi nos conseils, vous devriez à cet instant avoir envie de tester l'application sur un appareil physique réel ! Nous n'insisterons jamais assez : aucun simulateur, aussi parfait soit-il, ne remplacera jamais l'expérience sur un appareil physique. Depuis Xcode et Eclipse, ou grâce à la commande `cordova run`, déployez votre application sur votre iPhone, iPad ou encore Galaxy Note ou Nexus S.

Voilà, vous êtes maintenant prêt à construire votre première application ! Vous avez toutes les cartes en mains pour commencer à travailler. Nous avons toutefois rassemblé quelques conseils et astuces supplémentaires.

Architecture de l'application

config.xml, le fichier de configuration

Le fichier de configuration de notre application, `config.xml`, est relativement simple, regardons ensemble les lignes importantes qui le composent.

```
<?xml version="1.0" encoding="utf-8"?>

<widget id="com.cordobook.app" version="1.0.0" xmlns="http://www.w3.org/ns/widgets"
xmlns:cdv="http://cordova.apache.org/ns/1.0">
  <name>Rappelle-toi</name>
  <description>L'application de démonstration du livre</description>
  <author email="sebastien.pittion@fingerproof.net" href="http://fingerproof.net">
      Sébastien Pittion
  </author>
  <content src="index.html" />
```

Nous spécifions que la page d'accueil de l'application est la page `index.html`.

```
  <access origin="*" />
  <preference name="DisallowOverscroll" value="true"/>
  <preference name="UIWebViewBounce" value="false" />
  <preference name="Orientation" value="portrait" />
  <preference name="AutoHideSplashScreen" value="false" />
  <preference name="StatusBarOverlaysWebView" value="true" />
  <preference name="deployment-target" value="7.0" />
```

À travers plusieurs préférences, nous précisons qu'il n'est pas possible de scroller au-delà de la page, que le *bouncing* de la vue (c'est-à-dire la possibilité de scroller au-delà des limites du contenu, faisant revenir celui-ci en place comme un élastique) est désactivé, que l'application est

Copyright © 2015 Eyrolles.

disponible seulement en portrait, que l'écran de chargement ne disparaît pas seul, que l'application dispose d'une barre de statut qui chevauche la WebView ou encore que la version minimale d'iOS demandée est 7.0.

```
</widget>
```

Les fichiers CSS

Alors que la sémantique de l'application est réalisée par le biais de fichiers HTML, sa présentation est personnalisée à travers des fichiers CSS. Dans le cas de notre application de démonstration, ceux-ci sont organisés dans le dossier `www/css`, et nos fichiers sont de quatre types.

- Les fichiers CSS externes, que nous utilisons pour nous simplifier le travail, comme la police FontAwesome pour les pictogrammes `(font-awesome.min.css)` ou la police Pacifico pour le texte `(font-pacifico.css)`. À noter que le dossier CSS contient le fichier CSS des polices, pas le fichier source de la police elle-même, situé dans le dossier `fonts`.
- Le fichier principal `main.css` qui contient la structure de base de l'application, avec notamment le style de la balise html, ou encore les styles généraux appliqués à tous les éléments (`*`).
- Les fichiers de style de chaque composant de l'application : `bar.css`, `button.css`, `content.css`, `form.css`, `header.css`, `menu.css`, `modal.css`, `timeline.css` ou encore `view.css`. En effet, l'interface de l'application a été découpée en composants classiques, isolés, facilement maintenables et réutilisables dans d'autres projets si besoin. De plus, les sélecteurs CSS utilisés suivent la méthodologie BEM (*Block Element Modifier*, https://bem.com/method/) qui possède bien des avantages, encore une fois dans un but de maintenabilité et de séparation des responsabilités.
- Le fichier du thème, `theme.css`, qui contient le style lié au thème, c'est-à-dire principalement les couleurs des éléments, les ombres, etc. Séparer la mise en page (layout) du thème permet de ne changer facilement que ce qui est nécessaire en éliminant le plus de bruit possible.

Les polices d'écriture

Nous trouvons également dans le dossier `www` un dossier `fonts`, contenant les fichiers sources des polices que nous utilisons : FontAwesome et Pacifico[2].

Le fichier index.html

Comme dans n'importe quel site web, le fichier `index.html` constitue par défaut le point d'entrée de l'application. Sa structure est très proche de celle d'un site web. Regardons de plus près son contenu :

```
<!DOCTYPE html>
```

2. http://www.fontsquirrel.com/fonts/pacifico

Copyright © 2015 Eyrolles.

Cette balise est indispensable pour préciser le type du document, ici un document HTML 5 :

```
<html lang="fr">
  <head>
    <meta charset="utf-8">
```

La balise `charset` précise l'encodage du document :

```
    <title>Rappelle-toi</title>
    <meta name="viewport" content="initial-scale=1, user-scalable=no">
```

La balise `viewport` règle le comportement du *viewport* (la fenêtre de visualisation de l'application). Dans notre cas, ces options permettent un affichage du contenu à l'échelle 1:1 sur l'écran de l'appareil et empêchent l'utilisateur de zoomer.

```
    <link rel="stylesheet" href="css/font-awesome.min.css">
    <link rel="stylesheet" href="css/font-pacifico.css">
    <link rel="stylesheet" href="css/main.css">
```

Ces balises `link` permettent d'inclure les fichiers CSS de l'application. Il est généralement préférable de le faire le plus tôt possible pour éviter le FOUC *(Flash Of Unstyled Content)*, passage visible des styles de base du navigateur aux styles personnalisés.

```
  </head>
  <body class="app">
    <!-- Timeline view -->
    <section class="view" id="timeline">
      [Contenu de la vue Timeline]
    </section>

    <!-- Picture modal view -->
    <section class="view modal" id="picture">
      [Contenu de la vue Picture]
    </section>

    <!-- Video modal view -->
    <section class="view modal" id="video">
      [Contenu de la vue Video]
    </section>

    <!-- Text modal view -->
    <section class="view modal" id="text">
      [Contenu de la vue Text]
    </section>

    <!-- Location modal view -->
    <section class="view modal" id="location">
       [Contenu de la vue Location]
    </section>
```

Copyright © 2015 Eyrolles.

Voici comment insérer le code HTML qui constitue chaque vue.

```
<!-- Templates -->
<script id="imageTemplate" type="html/template">
 [Template de représentation d'une image]
</script>
<script id="videoTemplate" type="html/template">
 [Template de représentation d'une vidéo]
</script>
<script id="textTemplate" type="html/template">
  [Template de représentation d'un texte]
</script>
<script id="locationTemplate" type="html/template">
  [Template de représentation d'une géolocalisation]
</script>
<script id="helpTemplate" type="html/template">
  [Template d'une entrée dans l'aide]
</script>
<script id="timelineTemplate" type="html/template">
  [Template de représentation d'une entrée de la timeline]
</script>
```

Vous devez ensuite déclarer les différents templates, c'est-à-dire la représentation de chaque type d'entrées possibles dans la timeline. Ce code HTML ne sera ainsi affiché qu'en temps voulu par le biais de code JavaScript.

```
<!-- Scrips imports -->
<script src="cordova.js"></script>
<script src="js/lib/fastclick.js"></script>
<script src="js/data.js"></script>
<script src="js/utils.js"></script>
<script src="js/views/modal.js"></script>
<script src="js/views/timeline.js"></script>
<script src="js/views/picture.js"></script>
<script src="js/views/video.js"></script>
<script src="js/views/text.js"></script>
<script src="js/views/location.js"></script>
<script src="js/app.js"></script>
<script src="js/main.js"></script>
```

L'import des fichiers JavaScript s'effectue en dernier sur la page. Le fichier `cordova.js` est importé en premier, c'est Cordova qui le placera à la racine du dossier `www`. Il doit être référencé même s'il n'est pas présent lors du développement dans le navigateur web (provoquant une simple erreur de chargement). Cette erreur disparaîtra après son ajout par Cordova lors de la compilation de l'application.

```
  </body>
</html>
```

Copyright © 2015 Eyrolles.

Le fichier main.js

Ce fichier est le point d'entrée JavaScript de l'application, et c'est lui qui va lancer l'initialisation de l'application. Son code est volontairement très simple :

```
cordova.require('app').init();
```

Le fichier app.js

Le fichier `main.js` que nous venons de voir n'est en réalité chargé que lors du lancement de la fonction `init()` présente dans le fichier `app.js` dont voici le contenu. La fonction principale de ce dernier est d'initialiser chacune des vues de l'application.

```
// The app main module, exports an 'init' method to start the app.
// Cordova modules pretty much work like AMD modules (see RequireJS)
// thought without including any loading feature (scripts still need
// to explicitly be imported in 'index.html' using a 'script' tag).
// The main advantages of using those modules are easy namespacing
// and free closures preventing not needed global variables.
cordova.define('app', function (require, exports) {
  'use strict';

  // Import dependencies (Cordova modules).
  var utils = require('app.utils');
  var timelineView = require('app.views.timeline');
  var pictureView = require('app.views.picture');
  var videoView = require('app.views.video');
  var textView = require('app.views.text');
  var locationView = require('app.views.location');

  function onDeviceReady() { navigator.splashscreen.hide(); }

  // The app entry point. Of course we'll only call this function once.
  function init() {
    utils.on(document, 'deviceready', onDeviceReady);
    FastClick.attach(document.body);

    // Initialize views passing their associated root DOM element.
    // At this point, since scripts are inserted right before the
    // 'body' closing tag, we can query previous DOM elements without
    // waiting for the DOM to be ready. On the contrary, we could just
    // have moved those lines inside the 'onDeviceReady' callback.
    timelineView.init(utils.element('#timeline'));
    pictureView.init(utils.element('#picture'));
    videoView.init(utils.element('#video'));
    textView.init(utils.element('#text'));
    locationView.init(utils.element('#location'));
  }

  // Define the interface that will be accessible to other modules.
  exports.init = init;
});
```

Copyright © 2015 Eyrolles.

Ajouter une géolocalisation

Le fonctionnement de l'écran de l'application (la vue) qui va permettre d'ajouter sa localisation à notre journal est décrit dans son propre fichier JavaScript. Pour plus de clarté, nous vous présentons ci-après le code lié à l'utilisation du plug-in. Référez-vous au code disponible en ligne pour accéder au contenu complet du fichier.

```
function onLocationComplete() { locateElement.disabled = false; }

function onLocationError(error) {
  modal.showError(app.errors.generic + error.message);
  onLocationComplete();
}

// A possible improvement here would be to save the Google image
// file to the application storage directory and synchronize
// its lifecycle with the one of its associated entry.
function onLocationSuccess(position) {
  location = position.coords.latitude + ',' + position.coords.longitude;
  var html = utils.template(app.templates.location, format());
  // Show a preview in the dedicated DOM element.
  previewElement.innerHTML = html;
  var imageElement = utils.element('.j-map', previewElement);
  utils.on(imageElement, 'load', onLocationComplete);
  utils.on(imageElement, 'error', onLocationError);
}

function onLocateCLick() {
  // Temporarily disable the button to prevent spamming.
  locateElement.disabled = true;
  // Use the Cordova Geolocation plugin to get the user's geolocation.
  navigator.geolocation.getCurrentPosition(
    onLocationSuccess,
    onLocationError,
    { enableHighAccuracy: true }
  );
}
```

Ajouter une photo

Vous trouverez ici le code de la vue permettant l'ajout d'une image à notre timeline.

```
function onTakeComplete() { takePictureElement.disabled = false; }

function onTakeError(message) { // TODO: do nothing if user cancelled
  modal.showError(app.errors.generic + message);
  onTakeComplete();
}
```

Copyright © 2015 Eyrolles.

```
// A possible improvement here would be to move the image
// file to the application storage directory and synchronize
// its lifecycle with the one of its associated entry.
function onTakeSuccess(url) {
  pictureURL = url;
  var html = utils.template(app.templates.image, format());
  // Show a preview in the dedicated DOM element.
  previewElement.innerHTML = html;
  var imageElement = utils.element('.j-image', previewElement);
  utils.on(imageElement, 'load', onTakeComplete);
  utils.on(imageElement, 'error', onTakeError);
}

function onTakePictureClick() {
  clean();
  // Temporarily disable the button to prevent spamming.
  takePictureElement.disabled = true;
  // Use the Cordova Camera plugin to take a picture.
  navigator.camera.getPicture(onTakeSuccess, onTakeError, {
    destinationType: Camera.DestinationType.FILE_URI,
    sourceType: Camera.PictureSourceType.CAMERA,
    quality: 50
  });
}
```

Ajouter une vidéo

La vue permettant l'ajout d'une vidéo à notre timeline dispose également d'un fichier JavaScript dédié, le voici :

```
function onTakeComplete() { takeVideoElement.disabled = false; }

function onTakeError(error) {
  // Do not show an error dialog box if the user cancelled the take.
  var notCancelled = error.code !== CaptureError.CAPTURE_NO_MEDIA_FILES;
  if (notCancelled) { modal.showError(app.errors.generic + error.message); }
  onTakeComplete();
}

// A possible improvement here would be to move the video
// file to the application storage directory and synchronize
// its lifecycle with the one of its associated entry.
function onTakeSuccess(files) {
  // As playing local videos on Android is tricky, we need to
  // get the actual file entry URL on the device (using the
  // Cordova File plugin) to make it work as expected.
  window.resolveLocalFileSystemURL(files[0].fullPath, function (fileEntry) {
    videoURL = fileEntry.toURL();
    var html = utils.template(app.templates.video, format());
    // Show a preview in the dedicated DOM element.
```

Copyright © 2015 Eyrolles.

```
    previewElement.innerHTML = html;
    onTakeComplete();
  }, onTakeError);
}

function onTakeVideoClick() {
  // Temporarily disable the button to prevent spamming.
  takeVideoElement.disabled = true;
  var options = { duration: 10 };
  // Use the Cordova Media Capture plugin to record a video clip.
  navigator.device.capture.captureVideo(onTakeSuccess, onTakeError, options);
}
```

Bonnes pratiques et astuces

À mesure de vos développements, vous allez vous-même découvrir les bonnes pratiques et trouver des astuces pour améliorer votre application et vos processus de travail. Voici toutefois quelques généralités.

Un projet bien planifié est à moitié fait

Il peut être tentant d'installer Cordova et de tout de suite se lancer à corps perdu dans le code de votre application. Mais comme tout projet informatique, ou projet en général, une planification en amont permet de gagner beaucoup de temps, et de ne pas passer à côté de points essentiels. Nous avons vu précédemment les étapes « classiques » de préparation, de la définition du concept jusqu'à l'identification des plug-ins et des plates-formes cibles.

Tester, tester, tester

Nous n'allons pas encore le répéter, mais il est primordial de tester sur un appareil physique le plus tôt possible ! Pensez dès le départ à vérifier le fonctionnement de la brique Cordova. Le meilleur test est celui du `deviceReady`, autrement dit celui qui consiste à vérifier que le démarrage de Cordova se déroule correctement en effectuant un traitement, comme une `alert()`, dans la fonction `deviceReady`.

Pour cela, ajoutez le code suivant dans votre fichier `index.html`.

```
<script>
    var onDeviceReady = function () {
      alert('Cordova a démarré avec succès');
    }
    document.addEventListener('deviceready', onDeviceReady);
</script>
```

Si le démarrage de Cordova se passe bien, vous verrez l'alerte quelques secondes après le démarrage de votre application. En effet, l'alerte n'apparaît que si l'événement `deviceReady` survient, attestant du démarrage de Cordova.

Copyright © 2015 Eyrolles.

Être prêt à reconstruire le projet à tout moment

Si vous avez de l'expérience en tant que développeur, vous savez que tout ne fonctionne pas toujours comme on l'espérerait. Il arrive notamment qu'un projet Cordova, suite à l'ajout et au retrait de divers plug-ins par exemple, se retrouve dans un état instable, avec une compilation qui ne fonctionne plus. Dans ce cas, n'hésitez pas à reconstruire le projet entièrement. En principe, vous n'avez généralement pas à modifier le contenu du dossier `platforms`, et toute votre configuration s'effectue via le fichier `config.xml` ou encore l'utilisation de hooks (voir chapitre 5).

De la même manière, conservez la liste des plug-ins utilisés. Vous serez peut-être amené à tous les réinstaller. Vous pouvez également utiliser `cordova save plugins --experimental`, ou encore créer un faux plug-in nommé dependencies comme dans notre application de démonstration.

Chercher à mutualiser et éviter les processus manuels

Un projet d'application Cordova est avant tout un projet de développement informatique. Toutes les bonnes pratiques du développement s'appliquent donc, avec en priorité la nécessité de mutualiser votre code. Ne dupliquez jamais de code, cherchez toujours à factoriser. Référez-vous à des ressources sur le développement web pour en savoir plus.

De la même manière, faites en sorte de n'avoir aucune tâche manuelle à effectuer. Dans l'idéal, tout doit être automatisé autant que faire se peut. Par exemple, le déploiement de vos icônes et écrans de chargement devrait s'effectuer via un hook (voir plus loin). Même chose si vous devez personnaliser le fichier de configuration Android (appelé `Manifest`). Si, à chaque exécution de la commande `cordova build`, vous devez effectuer des tâches manuelles, vous finirez un jour ou l'autre par oublier de le faire…

Copyright © 2015 Eyrolles.

5

Ajout de fonctionnalités spécifiques

L'ajout des fonctionnalités spécifiques à une application s'effectue notamment à l'aide de plug-ins Cordova.

Installer les plug-ins

Il existe plusieurs manières d'installer un plug-in. Nous allons les passer en revue.

Depuis un repository Cordova

C'est le mode que vous rencontrerez le plus souvent. Il consiste à installer le plug-in en spécifiant son nom dans le repository de plug-ins de Cordova, de cette façon :

```
cordova plugin add org.apache.cordova.device
```

Depuis un repository GitHub

Il peut arriver que le plug-in qui vous intéresse n'existe pas dans le repository officiel mais soit mis à disposition sur GitHub par un contributeur. La commande d'installation est alors la suivante :

```
cordova plugin add https://github.com/brodysoft/Cordova-SQLitePlugin
```

Copyright © 2015 Eyrolles.

Depuis un dossier en local

Si vous avez téléchargé les sources du plug-in, vous pouvez l'installer avec la commande suivante :

```
cordova plugin add /path/to/directory
```

Comme nous l'avons vu, vous pouvez parfois être amené à réinstaller les plug-ins. Pour éviter de télécharger la toute dernière version du plug-in officiel ou sur GitHub, que vous n'aurez pas testée, ce sera peut-être une bonne idée de faire référence à une copie locale des plug-ins. Vous avez ainsi une maîtrise totale de vos plug-ins et ne subirez pas les modifications des développeurs qui pourraient vous affecter.

Avec plugman

Lors de vos recherches de plug-ins, certains devront peut-être être installés avec plugman[1]. Pour installer ce dernier, utilisez la commande :

```
npm install -g plugman
```

La commande d'installation des plug-ins dispose de la signature suivante :

```
plugman --platform <ios|amazon-fireos|android|blackberry10|wp7|wp8> --project
<directory> --plugin <name|url|path> [--plugins_dir <directory>] [--www <directory>]
[--variable <name>=<value> [--variable <name>=<value> ...]]
```

Voici un exemple de son utilisation sous Windows :

```
plugman install --platform android --project F:\my17app\ --plugin https://git-wip-us.
apache.org/repos/asf/cordova-plugin-device.git
```

Vous pouvez aussi installer les plug-ins officiels en utilisant plugman, avec, par exemple, cette commande :

```
plugman --platform --project --plugin org.apache.cordova.battery-status
```

Manuellement

Si vous êtes familier avec la précédente version de Cordova (2.X) ou que vous lisez des articles de blogs ou des discussions y faisant référence, vous serez peut-être tenté d'installer les plug-ins à la main, en plaçant vous-même les fichiers aux bons endroits. Nous ne pouvons que vous déconseiller cette méthode qui mène généralement à des erreurs, des oublis et des complications.

Fonctionnement des plug-ins

Pour illustrer le fonctionnement des plug-ins, nous passons ici en revue plusieurs de ceux utilisés dans l'application, mais aussi ceux dont vous pourriez avoir besoin.

1. https://github.com/apache/cordova-plugman

Copyright © 2015 Eyrolles.

Device

Device met à disposition une variable globale permettant d'avoir des informations sur le matériel et le logiciel de l'appareil mobile. Son installation s'effectue ainsi :

```
cordova plugin add org.apache.cordova.device
```

Comme tous les plug-ins, l'information qu'il renferme n'est accessible qu'après l'événement `Device Ready`. Voici un exemple de code à placer dans votre JavaScript, destiné à afficher la version de Cordova utilisée.

```
document.addEventListener('deviceready', onDeviceReady, false);
function onDeviceReady() {
    console.log(device.cordova);
}
```

L'utilisation du plug-in Device permet l'accès à la variable globale `device`, dont les attributs disponibles sont :

- `device.cordova` : pour connaître la version de Cordova ;
- `device.model` : pour savoir le nom du modèle d'appareil – cette valeur est définie par le constructeur ;
- `device.platform` : délivre le nom du système d'exploitation de l'appareil ;
- `device.uuid` : pour avoir l'UDID de l'appareil (*Universally Unique Identifier*, c'est-à-dire l'identifiant unique de l'appareil) ;
- `device.version` : donne la version du système d'exploitation de l'appareil.

Astuces

1. Sur iOS, l'UDID n'est pas unique, il change pour chaque application, pour chaque installation, voire lors de la mise à jour du système d'exploitation.
2. La propriété `device.platform` peut être utilisée pour appliquer des traitements différents suivant la plate-forme (iOS ou Android par exemple), en plus des possibilités offertes par le dossier `merges` et les hooks.

Consultez la documentation pour plus de détails[2].

SplashScreen

SplashScreen permet un contrôle plus fin de l'apparition et de la disparition de l'écran de chargement. Particulièrement pratique si vous souhaitez laisser un peu de temps à votre application pour démarrer, en affichant par exemple votre logo pendant quelques secondes. L'installation s'effectue avec la commande :

```
cordova plugin add org.apache.cordova.splashscreen
```

2. http://plugins.cordova.io/#/package/org.apache.cordova.device

Copyright © 2015 Eyrolles.

Le plug-in dispose de deux méthodes : `hide` et `show`.

La première, `hide`, sert à masquer l'écran de chargement.

```
navigator.splashscreen.hide();
```

La seconde, `show`, s'utilise au contraire pour l'afficher.

```
navigator.splashscreen.show();
```

Consultez la documentation officielle pour le configurer pour vos plates-formes cibles[3].

Camera

Camera permet à la fois de prendre des photos avec l'appareil photo, mais aussi de proposer à l'utilisateur de choisir une photo dans la galerie à disposition. Son installation s'effectue via la commande suivante :

```
cordova plugin add org.apache.cordova.camera
```

Une fois le plug-in installé, la méthode `navigator.camera.getPicture` sert à prendre une photo avec l'appareil, ou à récupérer une image dans la galerie de l'utilisateur. L'image sera au format base64 ou sous la forme d'un chemin vers l'image, sur le système de fichiers. Voici la signature de cette méthode :

```
navigator.camera.getPicture(cameraSuccess, cameraError, cameraOptions);
```

Les différents paramètres d'appel de cette méthode sont :

- `cameraSuccess` : le callback de succès retourne l'URL de l'image, ou son contenu en base-64 suivant la configuration de l'option `destinationType`.

```
function (imageData) {
    // Traitement de l'image
}
```

Astuce

Si, lors de vos phases de debug, vous êtes amené à utiliser la méthode `alert()` JavaScript dans l'un des callbacks, notez que sur iOS vous devrez entourer cet appel pour qu'il puisse fonctionner :

```
setTimeout(function () {
    // code de debug ici
}, 0);
```

3. http://plugins.cordova.io/#/package/org.apache.cordova.splashscreen

Copyright © 2015 Eyrolles.

Voici un exemple complet de prise de photo et récupération du contenu en base-64 :

```
navigator.camera.getPicture(onSuccess, onFail, {
    quality: 50,
    destinationType: Camera.DestinationType.DATA_URL
});

function onSuccess(imageData) {
    var image = document.getElementById('myImage');
    image.src = '' + imageData;
}

function onFail(message) {
    alert('Failed because: ' + message);
}
```

Et voyons maintenant un exemple complet de prise de photo et récupération de son URL :

```
navigator.camera.getPicture(onSuccess, onFail, {
    quality: 50,
    destinationType: Camera.DestinationType.FILE_URI
});

function onSuccess(imageURI) {
    var image = document.getElementById('myImage');
    image.src = imageURI;
}

function onFail(message) {
    alert('Failed because: ' + message);
}
```

- `cameraError` : le callback d'erreur fournit un message d'erreur qui peut être récupéré comme suit :

```
function (message) {
    // Traitement du message
}
```

- `cameraOptions` : liste d'options de configuration de la méthode `navigator.camera.getPicture` ;

  Le paramètre `cameraOptions` est un objet pouvant contenir les propriétés suivantes :

  – `quality` : qualité de l'image sauvegardée, entre 0 et 100, 100 étant la qualité maximum, et 50 étant la qualité par défaut ;

  – `destinationType` : choix du format de retour de l'image. Il y a trois valeurs possibles : `Camera.DestinationType.DATA_URL` (contenu de l'image en base-64), `Camera.DestinationType.FILE_URI` (URL de l'image sur le système de fichiers local) et `Camera.DestinationType.NATIVE_URI` (URL native de l'image, `assets-library://` sur iOS et `content://` sur Android) ;

Copyright © 2015 Eyrolles.

- `sourceType` : source de l'image. Trois valeurs sont possibles : `Camera.PictureSourceType.PHOTOLIBRARY` (galerie photos de l'appareil), `Camera.PictureSourceType.CAMERA` (prise de la photo avec l'appareil photo), `Camera.PictureSourceType.SAVEDPHOTOALBUM` (album photo sauvegardé par l'utilisateur) ;
- `allowEdit` : permission à l'utilisateur de modifier l'image avant de la valider ;
- `encodingType` : encodage de l'image retournée (`Camera.EncodingType.JPEG` ou `Camera.EncodingType.PNG`) ;
- `targetWidth` : largeur de l'image retournée, à utiliser avec `targetHeight` ;
- `targetHeight` : hauteur de l'image retournée, à utiliser avec `targetWidth` ;
- `mediaType` : sélection du type de média que l'utilisateur peut choisir dans le cas où `PictureSourceType` est `PHOTOLIBRARY` ou `SAVEDPHOTOALBUM`. Trois valeurs sont possibles : `Camera.MediaType.PICTURE`, `Camera.MediaType.VIDEO`, `Camera.MediaType.ALLMEDIA` ;
- `correctOrientation` : corrige l'orientation de l'image pour correspondre à l'orientation de l'appareil prenant la photo ;
- `saveToPhotoAlbum` : sauvegarde l'image dans l'album de l'utilisateur après la capture ;
- `cameraDirection` : choix de la camera à utiliser, entre la camera frontale et la caméra arrière. Deux valeurs possibles : `Camera.Direction.BACK`, `Camera.Direction.FRONT`.

Astuce

Sur des appareils récents, la photo peut être de grande qualité, et donc de taille importante. Pour éviter tout problème de mémoire (la mémoire vive disponible étant fortement limitée sur un appareil mobile), préférez la manipulation de l'image par URL (`Camera.destinationType` à `FILE_URI`) plutôt que de ses données elles-mêmes (`Camera.destinationType` à `DATA_URL`). Sinon, vous pouvez également réduire les dimensions et la qualité de l'image finale à l'aide des options dédiées.

Geolocation

Geolocation fournit des informations sur la position de l'appareil, comme la latitude ou la longitude. Ces valeurs sont obtenues en utilisant diverses sources d'information : le GPS, les réseaux Wi-Fi et Bluetooth, l'adresse IP, etc.

Astuce

Pour des raisons de protection de la vie privée, considérez l'affichage d'un pop-up d'alerte avant la récupération de la position GPS, afin d'informer l'utilisateur de que vous souhaitez en faire.

Le plug-in s'installe avec la commande suivante :

```
cordova plugin add org.apache.cordova.geolocation
```

Copyright © 2015 Eyrolles.

Il contient trois fonctions principales :

- `getCurrentPosition` : renvoie la position de l'appareil sous forme d'un objet `Position` au callback de succès, ou appelle le callback d'erreur en cas de problème. La méthode accepte un ensemble d'options pour un paramétrage plus fin.

```
navigator.geolocation.getCurrentPosition(geolocationSuccess,
                                         [geolocationError],
                                         [geolocationOptions]);
```

Voici un exemple d'utilisation de cette méthode issue de la documentation officielle, vous permettant du même coup de découvrir les propriétés disponibles dans l'objet `Position` :

```
function onSuccess(position) {
    alert('Latitude: ' + position.coords.latitude + '\n' +
          'Longitude: ' + position.coords.longitude + '\n' +
          'Altitude: ' + position.coords.altitude + '\n' +
          'Accuracy: ' + position.coords.accuracy + '\n' +
          'Altitude Accuracy: ' + position.coords.altitudeAccuracy  + '\n' +
          'Heading: ' + position.coords.heading + '\n' +
          'Speed: ' + position.coords.speed + '\n' +
          'Timestamp: ' + position.timestamp + '\n');
}

function onError(error) {
    alert('code: '    + error.code    + '\n' +
          'message: ' + error.message + '\n');
}

navigator.geolocation.getCurrentPosition(onSuccess, onError);
```

L'objet `Position` contient les propriétés suivantes :
  - `latitude` : latitude de l'appareil ;
  - `longitude` : longitude de l'appareil ;
  - `altitude` : altitude de l'appareil en mètre ;
  - `accuracy` : précision de la latitude et longitude en mètres ;
  - `altitudeAccuracy` : précision de l'altitude en mètres (non supporté sur Android) ;
  - `heading` : direction de déplacement, en degrés comptés dans le sens des aiguilles d'une montre à partir du Nord véritable ;
  - `speed` : vitesse courante en mètres par secondes.

- `watchPosition` : surveille la position à intervalles réguliers, et appelle le callback de succès lorsque celle-ci change. L'appel de la méthode renvoie un identifiant à utiliser pour arrêter la surveillance avec la méthode `navigator.geolocation.clearWatch`.

```
var watchId = navigator.geolocation.watchPosition(geolocationSuccess,
[geolocationError], [geolocationOptions]);
```

Copyright © 2015 Eyrolles.

Voici un exemple complet issu de la documentation officielle :

```
function onSuccess(position) {
    var element = document.getElementById('geolocation');
    element.innerHTML = 'Latitude: ' + position.coords.latitude + '<br />' + 'Longitude: '
+ position.coords.longitude + '<br />' + '<hr />' + element.innerHTML;
}

function onError(error) {
    alert('code: '    + error.code    + '\n' +
          'message: ' + error.message + '\n');
}

var watchID = navigator.geolocation.watchPosition(onSuccess, onError, { timeout: 30000 });
```

`geolocationOptions`, `getCurrentPosition` et `watchPosition` acceptent trois options :

- `enableHighAccuracy` : demande à l'appareil de fournir la position la plus précise possible, en évitant notamment les positionnements via le réseau de téléphonie mobile (triangulation grâce à la position des antennes relais), mais plutôt en privilégiant le positionnement satellite. Les appareils Android 2.X nécessitent que cette valeur soit `true` pour retourner une position ;
- `timeout` : temps maximal en millisecondes entre l'appel de la méthode et son appel au callback de succès. Si ce dernier n'est pas appelé dans les temps, le callback d'erreur sera invoqué avec un paramètre d'erreur `PositionError.TIMEOUT` ;
- `maximumAge` : indique la durée maximale pendant laquelle la position en cache est utilisée.

- `clearWatch` : permet de stopper la surveillance d'une position à partir d'un identifiant. Il est important d'y faire appel une fois la surveillance non requise, par exemple pour éviter d'éventuelles fuites de mémoire :

```
navigator.geolocation.clearWatch(watchID);
```

Lors d'une erreur, le callback d'erreur reçoit un objet `PositionError` contenant un code d'erreur parmi la liste suivante :

- `PositionError.PERMISSION_DENIED` : l'utilisateur n'a pas autorisé l'application à récupérer sa position ;
- `PositionError.POSITION_UNAVAILABLE` : l'appareil n'a pas été en mesure de récupérer la position ;
- `PositionError.TIMEOUT` : l'appareil n'a pas été en mesure de récupérer une position dans le timeout autorisé par les options.

Dialogs

Ce plug-in permet l'utilisation de boîtes de dialogue natives. Pour afficher des alertes, l'utilisation de la fonction `alert()` en JavaScript n'est pas une bonne idée, son rendu faisant tout de suite penser à une application web. Préférez plutôt l'utilisation de fenêtres de dialogue natives ! Son installation s'effectue à l'aide de la commande :

```
cordova plugin add org.apache.cordova.dialogs
```

Copyright © 2015 Eyrolles.

Le plug-in dispose de quatre méthodes.

- `alert` : affiche une alerte simple et accepte deux paramètres obligatoires et deux paramètres facultatifs :
  - `message` : texte de l'alerte ;
  - `alertCallback` : callback appelé lorsque l'alert est fermée par l'utilisateur ;
  - `title` : titre de la boîte de dialogue (optionnel) ;
  - `buttonName` : texte du bouton unique affiché dans la boîte de dialogue (optionnel).

```
navigator.notification.alert(message, alertCallback, [title], [buttonName]);
```

Voici un exemple d'utilisation issu de la documentation officielle :

```
function alertDismissed() {
    // do something
}

navigator.notification.alert(
    'You are the winner!',
    alertDismissed,
    'Game Over',
    'Done'
);
```

- `confirm` : affiche une boîte de dialogue de confirmation. Contrairement à la méthode `alert`, la boîte de dialogue `confirm` affiche un bouton de validation et un bouton d'annulation.

```
navigator.notification.confirm(message, confirmCallback, [title], [buttonLabels]);
```

Les paramètres d'appel de cette méthode sont :

- le paramètre `confirmCallback`, qui prend en paramètre l'identifiant du bouton qui a été pressé (1 ou 2) ou la valeur 0 si la fenêtre a été fermée sans presser aucun bouton (par exemple, en cliquant à côté de la fenêtre sur Android) ;
- le paramètre `buttonLabels`, qui contient un tableau de chaînes de caractères contenant les textes des boutons.

  Voici un exemple d'utilisation issu de la documentation officielle :

```
function onConfirm(buttonIndex) {
    alert('You selected button ' + buttonIndex);
}

navigator.notification.confirm(
    'You are the winner!',
    onConfirm,
    'Game Over',
    ['Restart','Exit']
);
```

Copyright © 2015 Eyrolles.

- `prompt` : affiche une boîte de dialogue permettant à l'utilisateur d'entrer une valeur.

```
navigator.notification.prompt(message, promptCallback, [title], [buttonLabels],
[defaultText]);
```

Les paramètres d'appel de cette méthode sont :

– Le paramètre `promptCallback` fonctionne comme le paramètre `confirmCallback` vu précédemment, à l'exception du fait que la fonction qui y sera précisée disposera de la valeur entrée dans le champ de saisie en tant que paramètre.

– Le paramètre `defaultText` est optionnel et contient le texte par défaut à afficher dans le champ de saisie.

Voici un exemple issu de la documentation officielle :

```
function onPrompt(results) {
    alert('You selected button number ' + results.buttonIndex + ' and entered ' +
results.input1);
}

navigator.notification.prompt(
    'Please enter your name',
    onPrompt,
    'Registration',
    ['Ok','Exit'],
    'Jane Doe'
);
```

- `beep` : un peu différente des autres, cette méthode permet de jouer un son de bip. Elle prend en paramètre le nombre de répétitions du son.

```
navigator.notification.beep(times);
```

Afin d'implémenter ces méthodes dans votre code, pensez à consulter la dernière version de la documentation officielle, et plus particulièrement les sections spécifiques aux plates-formes que vous utilisez. En effet, il existe de nombreux cas particuliers en fonction des plates-formes, et la lecture de la documentation est encore une fois essentielle.

Network information

Network information permet de récupérer des informations sur l'état de la connexion de l'appareil mobile. Il s'installe avec la ligne suivante :

```
cordova plugin add org.apache.cordova.network-information
```

Copyright © 2015 Eyrolles.

Ce plug-in donne accès à l'objet `Connexion`, via le code `navigator.connection.type`, qui contient les informations sur l'état de la connexion. Cet objet peut prendre les valeurs suivantes :

- `Connection.UNKNOWN` ;
- `Connection.ETHERNET` ;
- `Connection.WIFI` ;
- `Connection.CELL_2G` ;
- `Connection.CELL_3G` ;
- `Connection.CELL_4G` ;
- `Connection.CELL` ;
- `Connection.NONE`.

Voici un exemple issu de la documentation officielle :

```
function checkConnection() {
    var networkState = navigator.connection.type;

    var states = {};
    states[Connection.UNKNOWN]  = 'Unknown connection';
    states[Connection.ETHERNET] = 'Ethernet connection';
    states[Connection.WIFI]     = 'WiFi connection';
    states[Connection.CELL_2G]  = 'Cell 2G connection';
    states[Connection.CELL_3G]  = 'Cell 3G connection';
    states[Connection.CELL_4G]  = 'Cell 4G connection';
    states[Connection.CELL]     = 'Cell generic connection';
    states[Connection.NONE]     = 'No network connection';

    alert('Connection type: ' + states[networkState]);
}

checkConnection();
```

Astuce

iOS ne peut déterminer le type de connexion cellulaire (2G, 3G, 4G, générique). La réponse sera toujours `Connection.CELL`.

En complément, le plug-in permet également d'écouter deux événements et de rattacher des actions à la survenue de ces événements :

- `online` : apparaît lorsque l'appareil retrouve une connexion et l'accès à Internet ;

```
document.addEventListener('online', yourCallbackFunction, false);
```

- `offline` : apparaît lorsque l'appareil perd la connexion et que l'appareil n'est plus connecté à Internet.

```
document.addEventListener('offline', yourCallbackFunction, false);
```

Copyright © 2015 Eyrolles.

Battery Status

Battery Status permet d'obtenir des informations sur la batterie de l'appareil, et s'installe avec la commande suivante :

```
cordova plugin add org.apache.cordova.battery-status
```

Il ne contient pas de méthodes contrairement aux autres plug-ins, mais ajoute simplement trois événements qu'il est possible d'écouter.

- `batterystatus` : appelé lorsque le pourcentage de charge de la batterie change d'au moins un point, ou que l'appareil est branché ou débranché d'une source d'énergie. Deux paramètres sont renvoyés avec l'événement :
  - `level` : le pourcentage de charge de la batterie ;
  - `isPlugged` : un booléen indiquant si l'appareil est branché.

  Exemple issu de la documentation officielle :

```
window.addEventListener('batterystatus', onBatteryStatus, false);
function onBatteryStatus(info) {
    // Handle the online event
    console.log('Level: ' + info.level + ' isPlugged: ' + info.isPlugged);
}
```

Astuce

N'oubliez pas d'ajouter le listener pour cet événement après la survenue de l'événement `deviceReady`.

- `batterycritical` : fonctionne comme `batterystatus`, sauf qu'il survient lorsque la charge de la batterie atteint un niveau critique dépendant de chaque constructeur.
- `batterylow` : d'une manière similaire à `batterycritical`, il survient lorsque le niveau de la batterie atteint un niveau de charge faible dépendant de chaque constructeur.

Status Bar

Status Bar sert à personnaliser la barre de statut, c'est-à-dire la barre indiquant l'heure en haut de l'écran de l'appareil mobile. Il s'installe ainsi :

```
cordova plugin add org.apache.cordova.statusbar
```

Le plug-in se configure à l'aide de trois propriétés à placer dans le fichier `config.xml` :

- `StatusBarOverlaysWebView` : sur iOS 7, permet à la barre de statut de se superposer ou non à la WebView (booléen, `true` par défaut) ;

```
<preference name="StatusBarOverlaysWebView" value="true" />
```

- `StatusBarBackgroundColor` : sur iOS 7, règle la couleur de fond de la barre de statut (valeur hexadécimale) ;

```
<preference name="StatusBarBackgroundColor" value="#000000" />
```

Copyright © 2015 Eyrolles.

- `StatusBarStyle` : sur iOS 7, règle le style de barre de statut. Quatre styles sont disponibles : `default`, `lightcontent`, `blacktranslucent`, `blackopaque`.

```
<preference name="StatusBarStyle" value="lightcontent" />
```

Le plug-in met à disposition plusieurs méthodes pour contrôler la barre de statut.

- `overlaysWebView` permet de décider si la barre de statut passe au-dessus du contenu ou non sur iOS 7. Par exemple :

```
StatusBar.overlaysWebView(true);
```

- `styleDefault` permet de revenir au style par défaut pour la barre de statut. Par exemple :

```
StatusBar.styleDefault();
```

- `styleLightContent`, `styleBlackTranslucent`, `styleBlackOpaque` permettent d'utiliser le style `lightcontent`, `blacktranslucent` et `blackopaque` pour la barre de statut.
- `backgroundColorByName` permet de modifier la couleur de fond de la barre de statut. Les noms acceptés sont : `black`, `darkGray`, `lightGray`, `white`, `gray`, `red`, `green`, `blue`, `cyan`, `yellow`, `magenta`, `orange`, `purple`, `brown`. Par exemple :

```
StatusBar.backgroundColorByName('red');
```

Astuce

Sous iOS 7, la valeur `StatusBar.statusBarOverlaysWebView` doit être à `false` pour pouvoir modifier la couleur.

- `backgroundColorByHexString` permet de choisir la couleur à l'aide de sa valeur hexadécimale. Par exemple :

```
StatusBar.backgroundColorByHexString('#F3A456');
```

- `hide` permet de masquer la barre de statut. Par exemple :

```
StatusBar.hide();
```

- `show` permet d'afficher la barre de statut. Par exemple :

```
StatusBar.show();
```

- `isVisible` permet de savoir si la barre de statut est visible ou non. Par exemple :

```
if (StatusBar.isVisible) {
    // do something
}
```

InAppBrowser

Ce plug-in ouvre des URL directement depuis votre application sans que l'utilisateur ne soit redirigé vers Safari Mobile ou Chrome, par exemple. Il s'installe de la manière suivante :

```
cordova plugin add org.apache.cordova.inappbrowser
```

Copyright © 2015 Eyrolles.

Astuce

Gardez à l'esprit que le plug-in InAppBrowser possède le comportement d'une WebView classique, dépourvue des API Cordova.

L'installation du plug-in offre la possibilité d'utiliser la méthode `window.open`, qui permet d'ouvrir une page de votre choix dans le navigateur interne. Elle existe déjà en JavaScript, et ce sont en fait les paramètres utilisés qui vont définir si la page doit être ouverte dans le navigateur interne à l'application ou sortir de l'application pour ouvrir le navigateur par défaut de l'appareil mobile. Par la suite, nous utiliserons l'expression « navigateur interne » pour désigner l'instance d'`InAppBrowser`.

```
var ref = window.open(url, target, options);
```

Étudions de plus près les composantes de cette commande :

- `ref` : est la variable qui va accueillir l'instance du navigateur ;
- `url` : est l'URL à ouvrir. Pensez à appeler la méthode JavaScript native `window.encodeURI()` si cette URL contient des caractères spéciaux ;
- `target` : définit où va s'ouvrir la page. Les valeurs possibles sont :
  - `_self` : la page remplace le contenu de votre application (peu recommandé, car impossible de revenir en arrière sur iOS par exemple, et déroutant pour l'utilisateur). C'est la valeur par défaut ;
  - `_blank` : la page s'ouvre dans le navigateur interne sans quitter l'application ;
  - `_system` : la page s'ouvre dans le navigateur du téléphone, faisant quitter l'application à l'utilisateur.
- `options` : contient la liste des options, dont les paires clé-valeur sont séparées par des virgules, et ne doivent contenir aucun espace. Ces options sont :
  - `location` : yes pour conserver la barre d'adresse, no pour la masquer ;

    disponibles pour Android et iOS :
  - `closebuttoncaption` : texte du bouton Done ;
  - `hidden` : `yes` si vous voulez que le navigateur soit créé et charge la page, sans pour autant être visible à l'écran. L'événement `loadstop` est alors émis lorsque le chargement est complet. Omettez ce paramètre ou utilisez `no` pour avoir un comportement normal (par défaut : `no`) ;
  - `clearcache` : `yes` pour vider les cookies du navigateur interne avant son ouverture ;
  - `clearsessioncache` : `yes` pour vider le cache de sessions du navigateur interne avant son ouverture ;

    disponibles pour iOS uniquement :
  - `disallowoverscroll` : `yes` pour activer le « bouncing » (scroller au-delà de la page la fait rebondir jusqu'à sa position initiale ; par défaut : `no`) ;
  - `toolbar` : `yes` pour afficher la barre d'outils dans le navigateur interne (par défaut : `yes`) ;

Copyright © 2015 Eyrolles.

- `enableViewportScale` : `yes` pour éviter toute mise à l'échelle du viewport par le biais d'une balise meta (par défaut : `no`) ;
- `mediaPlaybackRequiresUserAction : yes` pour empêcher la lecture automatique de vidéos ou sons HTML 5 (par défaut : `no`) ;
- `allowInlineMediaPlayback` : `yes` pour autoriser le media playback HTML 5 (par défaut : `no`) ;
- `keyboardDisplayRequiresUserAction` : `yes` pour n'afficher le clavier virtuel que lorsque l'utilisateur donne le focus à un champ de formulaire (par défaut : `yes`) ;
- `suppressesIncrementalRendering` : `yes` pour attendre que le contenu du site web demandé soit entièrement chargé avant de l'afficher dans le navigateur interne (par défaut : `no`) ;
- `presentationstyle` : style de présentation du navigateur interne, au choix `pagesheet`, `formsheet` ou `fullscreen` (par défaut : `fullscreen`) ;
- `transitionstyle` : style de transition pour l'ouverture du navigateur interne, au choix `fliphorizontal`, `crossdissolve` ou `coververtical` (par défaut : `coververtical`) ;
- `toolbarposition` : position de la barre d'outils, au choix `top` ou `bottom` (par défaut : `bottom`).

L'objet retourné par la méthode `window.open` dispose de six méthodes supplémentaires :

- `addEventListener` : ajoute un listener sur un événement du navigateur interne, la syntaxe est simple :

```
ref.addEventListener(eventname, callback);
```

`ref` est l'objet retourné par la méthode `window.open`, `callback` est la méthode appelée lors de la survenue de l'événement précisé par son nom dans `eventname`, et pouvant prendre une des valeurs suivantes :

- `loadstart` : événement de début de chargement d'une URL dans le navigateur interne ;
- `loadstop` : événement de fin de chargement d'une URL dans le navigateur interne ;
- `loaderror` : événement d'erreur rencontrée lors du chargement d'une URL dans le navigateur interne ;
- `exit` : événement de fermeture du navigateur Internet.

Par exemple :

```
var ref = window.open('http://apache.org', '_blank', 'location=yes');
ref.addEventListener('loadstart', function (event) { alert(event.url); });
```

Lors de la survenue des événements, un objet contenant les propriétés suivantes est passé à la fonction `callback` :

- `type` : le nom de l'événement ;
- `url` : l'URL chargée dans le navigateur interne ;
- `code` : le code d'erreur si le type d'événement est `loaderror` ;
- `message` : le message d'erreur si le type d'événement est `loaderror`.

Copyright © 2015 Eyrolles.

- `removeEventListener` : supprime un écouteur précédemment ajouté au navigateur interne.

```
ref.removeEventListener(eventname, callback);
```

- `close` : ferme le navigateur interne.

```
ref.close();
```

- `show` : affiche le navigateur interne qui avait été ouvert avec l'option `hidden`.

```
var ref = window.open('http://apache.org', '_blank', 'hidden=yes');
// some time later...
ref.show();
```

- `executeScript` : injecte et exécute du code JavaScript dans la page affichée par le navigateur interne. Par exemple :

```
var ref = window.open('http://apache.org', '_blank', 'location=yes');
ref.addEventListener('loadstop', function () {
    ref.executeScript({ file: 'myscript.js' });
});
```

Le paramètre `details` permet de spécifier soit une URL vers le fichier à charger, soit le code JavaScript lui-même. La fonction `callback` sera appelée après l'exécution du JavaScript.

- `insertCSS` : injecte du CSS dans la page affichée par le navigateur interne. Par exemple :

```
var ref = window.open('http://apache.org', '_blank', 'location=yes');
ref.addEventListener('loadstop', function () {
    ref.insertCSS({ file: 'mystyles.css' });
});
```

La méthode accepte les mêmes paramètres que `executeScript`, à savoir un paramètre `details` contenant soit l'URL soit le code CSS lui-même, et une méthode `callback` appelée après l'injection.

Utilisation du répertoire merges

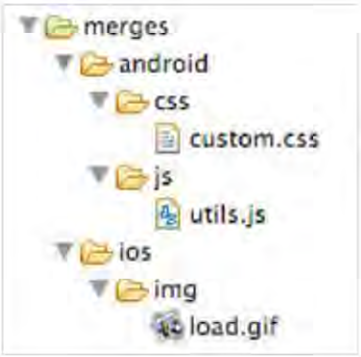
Le dossier `merges`, placé à la racine du projet Cordova, permet de spécifier des fichiers propres à une plate-forme donnée. Par exemple :

- proposer un CSS différent pour Android ou iOS ;
- proposer un écran de chargement différent pour Android ou iOS ;
- proposer une animation de chargement différente pour Android ou iOS.

Le dossier `merges` est composé d'un dossier par plate-forme cible, chacun respectant l'architecture du dossier `www`, et venant remplacer son contenu par les fichiers qu'il contient. Par exemple, considérons l'architecture illustrée par la figure 5-1.

Copyright © 2015 Eyrolles.

Figure 5-1
Dossier merges



Après l'exécution de la commande `cordova build ios`, le fichier `merges/ios/mg/load.gif` va venir remplacer le fichier `www/img/load.gif` s'il existe, être copié à cet endroit sinon.

Même chose avec l'exécution de la commande `cordova build android` qui aura pour effet de remplacer les fichiers `www/css/custom.css` et `www/js/utils.js`.

Lorsque vous travaillez avec le dossier `merges`, gardez bien à l'esprit que vous ne verrez pas toujours son impact direct sur votre application pendant la phase de debug. Pour cela, vous devrez utiliser la commande `cordova serve` prévue à cet effet, ou au minimum la commande `cordova prepare`.

En effet, lorsqu'on ouvre simplement le fichier `index.html` dans Chrome pour le debug, le navigateur n'a aucune connaissance du dossier `merges` et de sa logique de fonctionnement. L'utilisation de la commande `cordova serve` crée un serveur web sur lequel vous pourrez choisir la plate-forme à tester ; le contenu du dossier `merges` aura été préalablement copié dans le dossier `www` que vous consulterez.

Utilisation des hooks

Le concept de *hook* (« crochet » en français) est utilisé par de nombreux frameworks de développement. Un hook est un script exécuté à une étape précise du déploiement ou de l'exécution de votre code. Lors de l'utilisation de la commande `cordova build` par exemple, votre code va subir l'exécution de plusieurs commandes comme `cordova prepare` ou `cordova compile`. Via les hooks, vous pouvez exécuter des scripts avant et après chacune des commandes Cordova, ce pour une personnalisation avancée et/ou une intégration facilitée à certaines méthodes de travail.

Voici quelques exemples d'utilisations parmi de multiples :

- déployer automatiquement les icônes et écrans de chargement de l'application ;
- supprimer une permission du Manifeste Android qui a été ajouté par un plug-in, sans changer le code source de celui-ci ;
- remplacer une clé de développement par une clé de production lors de la création d'une application de production ;
- supprimer des fichiers temporaires inutiles.

Copyright © 2015 Eyrolles.

L'ajout d'un hook s'effectue via la création d'un fichier contenant le code du script et qu'on place dans le dossier `hooks` à la racine du projet Cordova. La liste suivante présente l'ensemble des noms de dossiers qu'il est possible d'ajouter au dossier `hooks`. Il faudra ensuite placer vos scripts dans le dossier correspondant à la commande Cordova à laquelle ils seront « rattachés ». Au sein de chaque dossier, les scripts sont exécutés par ordre alphabétique. Ainsi, un script dans le dossier `after_emulate` sera exécuté après la commande `cordova emulate`.

```
after_build/
    after_compile/
    after_docs/
    after_emulate/
    after_platform_add/
    after_platform_rm/
    after_platform_ls/
    after_plugin_add/
    after_plugin_ls/
    after_plugin_rm/
    after_plugin_search/
after_plugin_install/
    after_prepare/
    after_run/
    after_serve/
    before_build/
    before_compile/
    before_docs/
    before_emulate/
    before_platform_add/
    before_platform_rm/
    before_platform_ls/
    before_plugin_add/
    before_plugin_ls/
    before_plugin_rm/
    before_plugin_search/
before_plugin_install/
before_plugin_uninstall/
    before_prepare/
    before_run/
    before_serve/
    pre_package/ <-- Windows 8 and Windows Phone only.
```

Astuce

Si l'ordre d'exécution de vos scripts au sein d'un même dossier a pour vous une quelconque importance, n'hésitez pas à préfixer leurs noms par des chiffres.

Autre exemple, un script placé dans le dossier `after_plugin_add` sera exécuté après l'ajout d'un plug-in, et `before_serve` le sera avant l'utilisation de la commande `cordova serve`.

Copyright © 2015 Eyrolles.

Hormis sous Windows, pensez à rendre vos fichiers de script exécutables via la commande suivante ou l'une de ses variantes (en vous référant à la documentation, comme vous l'avez vu en première partie de cet ouvrage) :

```
chmod 755 monscript.js
```

Chaque script est exécuté à partir de la racine du dossier du projet Cordova, et le premier argument passé en paramètre est justement le chemin vers la racine du projet. Vous pouvez également tirer parti des variables d'environnement suivantes dans vos scripts :

- `CORDOVA_VERSION` : la version de Cordova-CLI ;
- `CORDOVA_PLATFORMS` : la liste des plates-formes sur lesquelles doit logiquement s'appliquer le script, séparées par des virgules (par exemple : `android, ios`) ;
- `CORDOVA_PLUGINS` : la liste des plug-ins sur lesquels le script peut s'appliquer, séparés par des virgules (par exemple : `org.apache.cordova.file`, `org.apache.cordova.file-transfer`) ;
- `CORDOVA_HOOK` : le chemin vers le script en question ;
- `CORDOVA_CMDLINE` : la commande exacte ayant déclenché l'exécution du script (par exemple : `cordova run ios --emulate`).

Comme toute commande, votre script peut retourner une erreur, c'est-à-dire une valeur autre que 0 (par convention). Si tel est le cas, l'exécution de la commande `Cordova` associée sera annulée.

Note

Selon les systèmes d'exploitation et la nature de vos scripts, il peut être nécessaire de commencer ces derniers par une ligne précisant le programme capable de les interpréter :

```
#!/usr/bin/env [name_of_interpreter_executable]
```

Vous trouverez plus d'informations sur le dépôt officiel Cordova sur GitHub[4].

Exemples de hooks

De nombreux exemples sont disponibles en ligne[5]. Nous allons en étudier quelques-uns.

Déployer les icônes et écrans de chargement

Que ce soit pour Android, iOS ou toute autre plate-forme, votre application a besoin d'avoir une icône et accessoirement d'un écran de chargement pour être parfaite ! Tandis que l'icône s'affiche sur l'écran d'accueil d'un appareil mobile (le *springboard*), le splashscreen correspond à l'écran affiché lors du lancement de l'application. Chaque plate-forme dispose de ses propres prérequis en termes de nommage, format, extension et taille de fichiers. Ceux-ci évoluent parfois en même temps qu'un système d'exploitation ou d'appareils donnés. C'est notamment le cas pour iOS, dont les appareils sont toujours plus grands, plus larges ou de meilleure résolution.

4. https://github.com/apache/cordova-lib/blob/master/cordova-lib/templates/hooks-README.md
5. http://devgirl.org/2013/11/12/three-hooks-your-cordovaphonegap-project-needs

Copyright © 2015 Eyrolles.

Astuce

Pour connaître les attentes d'iOS en termes d'icônes et d'écran de chargement, référez-vous à la documentation officielle Apple[17]. Au moment de la soumission à l'App Store, si des ressources nécessaires sont manquantes, cela vous sera signalé.

Les icônes iOS sont hébergées dans le dossier `platforms/ios/MONAPP/Resources/icons`, `MONAPP` étant le nom de votre application. Le dossier des écrans de chargement est le dossier `splash`, localisé au même niveau que le dossier `icons`. Les icônes et les écrans de chargement Android sont quant à eux placés dans les dossiers `platforms/android/res/drawable/FORMAT`, `FORMAT` étant l'un des formats ci-dessous :

- `drawable-ldpi` ;
- `drawable-mdpi` ;
- `drawable-hdpi` ;
- `drawable-xhdpi` ;
- `drawable-xxhdpi` ;
- `drawable-land-ldpi` ;
- `drawable-land-mdpi` ;
- `drawable-land-hdpi` ;
- `drawable-land-xhdpi` ;
- `drawable-port-ldpi` ;
- `drawable-port-mdpi` ;
- `drawable-port-hdpi` ;
- `drawable-port-xhdpi`.

Chaque format correspond à une résolution (ldpi – LOW, mdpi – MEDIUM, hdpi – HIGH, xhdpi – EXTRA HIGH, xxhdpi – EXTRA EXTRA HIGH) et une orientation (normal - pas de préfixe, paysage - land, portrait - port). Au sein de chaque dossier se trouve un fichier `icon.png` et `splash.png`.

Astuces

1. Lors de la construction de votre projet exécutable dans Xcode, Cordova place des icônes et écrans de chargement par défaut dans ces dossiers. Ceux-ci correspondent en principe aux formats demandés par Apple, mais les règles peuvent avoir évolué depuis votre dernière mise à jour de Cordova.
2. Le nom du fichier correspondant à l'écran de chargement Android peut être spécifié via la ligne suivante dans le fichier `config.xml` :

```
<preference name="SplashScreen" value="splash" />
```

6. https://developer.apple.com/library/ios/navigation/

Copyright © 2015 Eyrolles.

Comme nous l'avons vu précédemment, nous vous conseillons de limiter les interventions manuelles sur vos projets Xcode et Android. Vous pourrez ainsi supprimer et reconstruire ces projets sans vous soucier d'une quelconque liste de choses à faire. Notre hook va donc permettre de récupérer les icônes et écrans de chargement dans un dossier que l'on aura créé sous `www`, et de les placer dans les dossiers respectifs Android et iOS.

Voici le code complet de notre hook, stocké dans le fichier `01_icons_splashs.js`, suivi d'un aperçu de notre dossier `www/res` :

```
#!/usr/bin/env node

var shell = require('shelljs');

shell.exec('cp -R ./www/res/screen/ios/* platforms/ios/MONAPP/Resources/splash');
shell.exec('cp -R ./www/res/icon/ios/* platforms/ios/MONAPP/Resources/icons');
shell.exec('cp ./www/res/icon/android/icon-96-xhdpi.png platforms/android/res/
drawable/icon.png');
shell.exec('cp ./www/res/icon/android/icon-72-hdpi.png platforms/android/res/drawable-
hdpi/icon.png');
shell.exec('cp ./www/res/icon/android/icon-36-ldpi.png platforms/android/res/drawable-
ldpi/icon.png');
shell.exec('cp ./www/res/icon/android/icon-48-mdpi.png platforms/android/res/drawable-
mdpi/icon.png');
shell.exec('cp ./www/res/icon/android/icon-96-xhdpi.png platforms/android/res/
drawable-xhdpi/icon.png');

shell.exec('cp ./www/res/screen/android/screen-xhdpi-portrait.png platforms/android/
res/drawable/splash.png');
shell.exec('cp ./www/res/screen/android/screen-hdpi-portrait.png platforms/android/
res/drawable-hdpi/splash.png');
shell.exec('cp ./www/res/screen/android/screen-ldpi-portrait.png platforms/android/
res/drawable-ldpi/splash.png');
shell.exec('cp ./www/res/screen/android/screen-mdpi-portrait.png platforms/android/
res/drawable-mdpi/splash.png');
shell.exec('cp ./www/res/screen/android/screen-xhdpi-portrait.png platforms/android/
res/drawable-xhdpi/splash.png');

shell.exec('rm -r platforms/ios/www/res');
shell.exec('rm -r platforms/android/assets/www/res');
```

Copyright © 2015 Eyrolles.

Figure 5-2
Dossier des ressources

Examinons en détail le code du hook :

```
#!/usr/bin/env node
```

Cette ligne spécifie l'interpréteur à utiliser. Nous utilisons ici du code Node.js.

```
var shell = require('shelljs');
```

Pour des raisons de confort et d'habitude, le reste du script a été écrit avec du code Bash classique. Notez que nous aurions tout à fait pu nous passer de Node.js et utiliser directement Bash comme interpréteur. Nous utilisons le module `shelljs` de NodeJS pour exécuter des instructions. Vous devrez l'installer avec la commande :

```
npm install [-g] shelljs
```

Plus d'informations sur la page GitHub du projet : https://github.com/arturadib/shelljs.

```
shell.exec('cp -R ./www/res/screen/ios/* platforms/ios/MONAPP/Resources/splash');
shell.exec('cp -R ./www/res/icon/ios/* platforms/ios/MONAPP/Resources/icons');
```

Nous avons stocké les icônes et écrans de chargement dans le dossier `res`, et nous copions le contenu des dossiers `splash` et `icons` d'iOS dans leurs dossiers respectifs. Pensez à remplacer `MONAPP` par le nom de votre projet.

```
shell.exec('cp ./www/res/icon/android/icon-96-xhdpi.png platforms/android/res/
drawable/icon.png');
```

Copyright © 2015 Eyrolles.

Nous déplaçons et renommons chaque icône et écran de chargement Android. Cette étape est à adapter en fonction de l'évolution des formats requis et des noms de fichiers.

```
shell.exec('rm -r platforms/ios/www/res');
shell.exec('rm -r platforms/android/assets/www/res');
```

Le dossier `res` qui a été déployé en même temps que le contenu du dossier `www` peut être supprimé dans les projets finaux.

Astuce

Les fichiers de splashscreens peuvent être particulièrement volumineux et par conséquent alourdir fortement l'application finale. Pensez à utiliser un logiciel de compression comme ImageOptim pour réduire leur poids sans perdre en qualité.

Placez ce fichier dans le dossier `hooks/after_prepare` et constatez son fonctionnement en exécutant la commande `cordova build android`, `cordova build ios`, voire simplement `cordova prepare`. Ce hook est un simple exemple, puisque Cordova a récemment introduit la notation `<splash>` et `<icon>` à son fichier de configuration, permettant d'effectuer exactement ce que ce hook permet ! Prenez-le donc comme une démonstration des possibilités offertes par les hooks, et comme un exemple d'utilisation de `shelljs`.

Remplacer les chaînes de caractères

Vous pouvez avoir besoin de modifier ou supprimer une chaîne de caractères dans les fichiers de l'application. Il peut être, par exemple, intéressant d'enlever un flag utilisé pour le debug de l'application, passant ainsi le code « en production », ou encore de corriger le comportement de certains plug-ins qui modifient le fichier `AndroidManifest.xml` en ajoutant des permissions parfois inadaptées.

Astuce

Le sujet des permissions Android est important. Lors de l'installation de l'application, Android demande aux utilisateurs d'autoriser l'accès à un certain nombre de fonctionnalités du téléphone. Si votre application demande l'accès à la géolocalisation ou aux contacts, alors qu'a priori elle n'en a pas besoin, l'utilisateur peut décider de ne pas faire confiance à votre application.

Nous avons créé un fichier `01_replace_text.js` que nous avons placé dans le dossier `hooks/after_build`. Ce hook est largement inspiré des exemples postés sur le blog d'Holly Schinsky[7].

Le contenu du fichier est le suivant :

```
#!/usr/bin/env node

var fs = require('fs');
var path = require('path');
```

7. http://devgirl.org/2013/11/12/three-hooks-your-cordovaphonegap-project-needs

Copyright © 2015 Eyrolles.

```
var rootdir = process.argv[2];

function replace_string_in_file(filename, to_replace, replace_with) {
    var data = fs.readFileSync(filename, 'utf8');
    var result = data.replace(new RegExp(to_replace, 'g'), replace_with);
    fs.writeFileSync(filename, result, 'utf8');
}

if (rootdir) {
    var filestoreplace = ['platforms/android/AndroidManifest.xml'];
    filestoreplace.forEach(function (val, index, array) {
        var fullfilename = path.join(rootdir, val);
        if (fs.existsSync(fullfilename)) {
            replace_string_in_file(fullfilename, '<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />', '');
        } else {
            console.log('missing: ' + fullfilename);
        }
    });
}
```

Il n'est pas nécessaire de détailler le code avec précision. Gardez simplement à l'esprit que vous devez lister les fichiers concernés dans la variable `filestoreplace` et configurer vos remplacements dans `replace_string_in_file`. Dans notre cas, nous supprimons l'autorisation `ACCESS_FINE_LOCATION` ajoutée par un plug-in et qui est bien trop intrusive (et inutilisée par le plug-in dans notre cas).

Remplacer les chaînes de caractères en fonction de la cible de déploiement

Le script que nous venons de voir permet de couvrir de nombreux cas où le remplacement d'une chaîne de caractères est nécessaire. Parfois cependant, vous pourriez être amené à remplacer une chaîne en fonction de la cible de votre déploiement : développement ou production. La situation se présente rapidement dès que vous utilisez des services tiers, comme Facebook Connect, par exemple. En effet, ceux-ci ont souvent des clés de développement et de production, et leur remplacement à la main n'est pas envisageable – ou en tout cas non recommandé, comme nous l'avons vu précédemment.

Adaptons donc le script vu précédemment pour prendre en compte une cible de déploiement. Notre fichier se nomme `01_replace_text_target.js`. Il est placé dans le dossier `hooks/before_prepare`.

```
#!/usr/bin/env node

var fs = require('fs');
var path = require('path');
var rootdir = process.argv[2];

function replace_string_in_file(filename, to_replace, replace_with) {
    var data = fs.readFileSync(filename, 'utf8');
    var result = data.replace(new RegExp(to_replace, 'g'), replace_with);
    fs.writeFileSync(filename, result, 'utf8');
}
```

Copyright © 2015 Eyrolles.

```
var target = 'dev';
if (process.env.TARGET) {
    target = process.env.TARGET;
}

if (rootdir) {
    var ourconfigfile = path.join(rootdir, 'hooks', 'config.json');
    var configobj = JSON.parse(fs.readFileSync(ourconfigfile, 'utf8'));

    var filestoreplace = ['config.xml'];
    filestoreplace.forEach(function(val, index, array) {
        var fullfilename = path.join(rootdir, val);
        if (fs.existsSync(fullfilename)) {
            replace_string_in_file(fullfilename, "com.urbanairship.in_production'
value='false", configobj[target].ua);
            replace_string_in_file(fullfilename, "com.urbanairship.in_production'
value='true", configobj[target].ua);
        } else {
            console.log('missing: ' + fullfilename);
        }
    });
}
```

Nous pouvons voir que le script fait usage de la variable d'environnement `TARGET` qui va contenir la cible qu'on souhaite déployer, dans notre cas `dev` ou `prod`. Ainsi, il est important d'exécuter la commande `build` de la manière suivante :

```
TARGET=prod cordova build ios
```

ou :

```
TARGET=dev cordova build ios
```

Le script fait également référence à un fichier de configuration, `config.json`, placé dans le dossier `hooks`.

```
{
    "dev": {
       "ua": "com.urbanairship.in_production' value='false"
    },
    "prod": {
       "ua": "com.urbanairship.in_production' value='true"
    }
}
```

Vous constaterez que ce script s'exécute à l'étape `before_prepare`, et modifie le contenu du fichier `config.xml` source, et non celui qui est déployé. Il aurait également été possible de mettre dans la variable `com.urbanairship.in_production` une valeur « repère » (autrement dit, une chaîne de caractères représentative) comme `UA_PRODUCTION_KEY`, puis de modifier le script pour remplacer cette valeur à l'étape `after_build` dans le fichier de configuration cible.

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

6

Debug du code

Dans un monde idéal, vous n'auriez probablement pas besoin de ce chapitre puisque votre code parfait fonctionnerait du premier coup. Malheureusement, la réalité est tout autre, et vous serez souvent obligé de débugger votre code. Autant que cela se fasse dans un environnement que vous maîtrisez.

Dans un navigateur

Dans le cadre d'un développement d'application Cordova, plus grande partie du développement a généralement lieu dans le navigateur web. En effet, même si votre application comporte quelques plug-ins, il est plus pratique de travailler sur les vues et la logique du code directement dans votre navigateur préféré, plutôt que de devoir constamment déployer dans un simulateur ou sur un appareil. Par la suite, nous utiliserons Google Chrome comme navigateur de référence.

Pour visualiser votre application dans un navigateur, il existe trois méthodes principales.

- La première méthode est la plus naturelle. Elle consiste à ouvrir le fichier `index.html` directement dans Chrome (accès par URL au fichier d'index directement sur le disque). La barre d'adresse prend alors la forme `file://chemin_du_dossier_index.html`. Cependant, cette solution présente plusieurs inconvénients. Par exemple, Chrome empêche l'exécution de requêtes XHR dans ce cas, rendant des frameworks comme Angular inutilisables. Avec cette méthode, vous pourriez également rencontrer des soucis concernant l'origine de vos requêtes Ajax. En effet, certaines API nécessitent que les appels soient émis en provenance d'une URL particulière, et refuse les origines « vides » comme ce serait le cas ici.

Copyright © 2015 Eyrolles.

- La deuxième méthode consiste à mettre en place votre propre serveur web, héberger le code de votre application sur votre serveur, et éventuellement y accéder à travers un hôte virtuel (virtual host). Vous pouvez ainsi accéder à votre application à travers une adresse du type http://monapplication.com. Mais cette méthode pose toutefois un souci. Rappelez-vous : votre fichier `index.html` doit référencer un fichier `cordova.js` qui sera ajouté lors du déploiement dans votre dossier `www`, qui sera lui-même personnalisé, suivant les plates-formes, avec le contenu du dossier merges. Or, avec cette méthode, vous ne visualisez pas l'application comme elle le serait sur iOS ou Android, et vous recevez systématiquement une erreur concernant le fichier `cordova.js` qui est manquant, car vous visualisez l'application avant son déploiement.
- La troisième option apporte une solution à ces limitations. La commande `cordova serve`, exécutée dans le dossier de votre projet, permet de déployer votre application sur un serveur local créé spécialement pour l'occasion par Cordova. Une fois la commande exécutée, rendez-vous sur http://localhost/8000 pour y découvrir des informations sur votre application (nom, versions, liste des plug-ins), mais également accéder à votre application telle qu'elle sera déployée suivant les plates-formes que vous avez ajoutées. De plus, le fichier `cordova.js` est automatiquement déployé, vous pouvez ainsi utiliser sans problème les fonctions comme `cordova.require` dans votre code JavaScript. Notez toutefois que l'utilisation des plug-ins ne reste possible qu'une fois votre code déployé sur un simulateur ou un appareil.

Figure 6-1
Écran d'accueil d'un projet après la commande "cordova serve"

Astuce

Si vous n'utilisez pas le dossier `merges` par exemple, la solution du serveur web personnel paraît la plus adaptée : pas besoin de lancer systématiquement la commande `cordova serve`, possibilité d'héberger sur un domaine particulier en cas de limitation de l'API en termes d'origine…

Copyright © 2015 Eyrolles.

Utiliser Chrome Dev tools

Comme nous l'avons précisé précédemment, cet ouvrage privilégie l'utilisation de Google Chrome comme navigateur de développement. Nous le verrons plus tard, le debug à distance d'une application Android ne peut s'effectuer qu'avec Chrome, tandis que le debug d'une application iOS ne peut s'effectuer qu'avec Safari. Notre choix s'est porté sur Chrome, qui utilise le même moteur qu'iOS et Android (WebKit), mais aussi en raison d'un outil très puissant qu'il propose : Google Chrome Dev Tools.

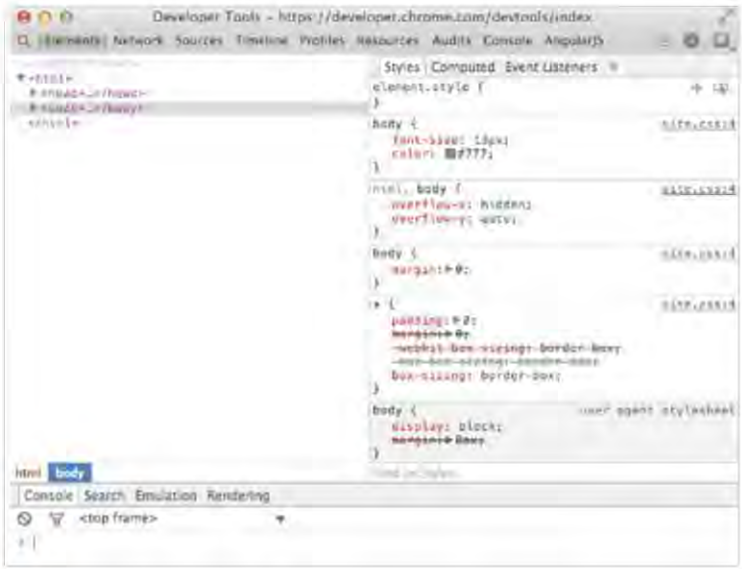
Google Chrome Dev Tools consiste en fait en une série d'outils à disposition des développeurs web, permettant d'explorer le contenu des pages web, le modifier, le débugger, l'auditer, le mesurer… Safari possède également un ensemble d'outils similaires.

Chrome Dev Tools est accessible de deux manières simples :

- en pressant les touches Cmd + Alt + i sur Mac, Ctrl + Shift + i ou F12 sur Windows ;
- en effectuant un clic droit sur un élément de la page et en choisissant Procéder à l'inspection de l'élément.

Une nouvelle fenêtre s'ouvre alors, contenant plusieurs onglets correspondant chacun à un outil. Nous allons parcourir rapidement les principaux outils que contient Chrome Dev Tools : Elements, Console, Emulation, Network, Sources, Profiles, Audits et Resources.

Figure 6-2
Google Chrome Dev Tools



Elements

Cette fenêtre permet d'explorer le DOM, c'est-à-dire la structure de la page et son contenu. C'est cet outil qui s'affiche lorsqu'on demande à inspecter un élément. Dans la fenêtre principale, il est possible de naviguer dans le DOM, qui se présente sous la forme d'un arbre de nœuds. On peut éditer le contenu des nœuds (en double-cliquant), déplacer les nœuds (en effectuant un cliquer-déplacer) ou encore supprimer un nœud (en le sélectionnant, puis en appuyant sur Backspace sur Mac ou Suppr sur Windows). Au survol d'un nœud, ce dernier se colore, vous permettant ainsi de voir où vous vous situez sur la page.

Copyright © 2015 Eyrolles.

Dans la seconde partie de la fenêtre, vous pouvez visualiser le style CSS appliqué au nœud actuellement sélectionné. L'onglet Styles contient l'ensemble des styles CSS appliqués à ce nœud, dans l'ordre dans lequel ils sont appliqués. L'onglet Computed contient les styles finaux appliqués, c'est-à-dire la dernière valeur ayant surchargé toutes les autres pour une propriété donnée. Dans l'onglet Styles, vous avez la possibilité de modifier les propriétés ou les valeurs, ou même d'ajouter de nouvelles propriétés ou de nouvelles classes.

Note

Que ce soit dans la fenêtre principale ou secondaire, toutes vos modifications ne sont que temporaires et disparaîtront au prochain rafraîchissement, à moins que vous ne répercutiez les modifications dans votre code.

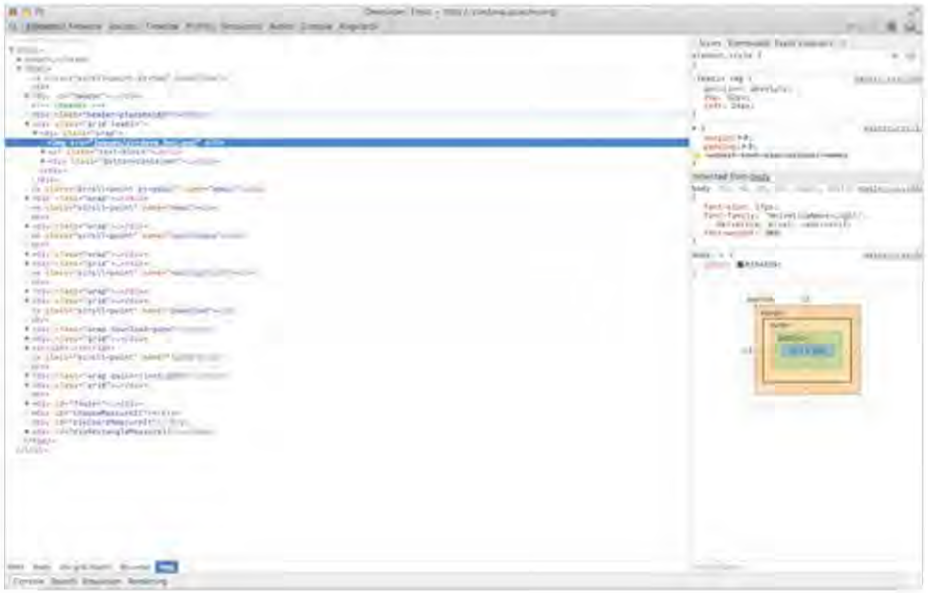


Figure 6-3
Onglet Elements de Google Chrome Dev Tools

Console

Cet outil correspond au huitième onglet de la fenêtre principale, mais fait également partie d'un volet « alternatif » qui s'ouvre et se ferme grâce à la touche Esc de votre clavier.

La console possède deux fonctions : afficher tous les messages de l'application (erreurs, avertissements, informations) et interagir avec l'application via du code JavaScript. Vous pouvez, par exemple, entrer `alert('Hello World')` ; dans la console, et l'alerte JavaScript apparaîtra dans votre application. C'est également le bon endroit pour tester vos sélecteurs CSS, si vous utilisez jQuery.

Figure 6-4
Onglet Console de Google Chrome Dev Tools

Copyright © 2015 Eyrolles.

Emulation

Emulation, présent dans le troisième onglet de la fenêtre alternative, comprend quatre sections :

- Device qui émule un appareil mobile, comme un iPad 4 ou un iPhone 5, directement dans Chrome ;
- Screen qui permet de choisir la taille de l'écran de l'appareil qu'on simule ;
- User Agents qui modifie le User Agent du navigateur, afin de le faire passer pour un appareil mobile particulier ;
- Sensors qui règle la valeur de plusieurs capteurs comme le GPS ou l'accéléromètre, ou encore d'activer la simulation des événements `touch`.

Figure 6-5
Onglet Emulation de Google Chrome Dev Tools

Network

Network est idéal pour visualiser en temps réel les requêtes effectuées par l'application, tant au niveau local pour aller chercher les images et fichiers, que distant pour exécuter des requêtes Ajax ou aller chercher du contenu externe. Prêtez attention au temps de chargement de chacun des fichiers pour optimiser les performances de votre application.

Figure 6-6
Onglet Network de Google Chrome Dev Tools

Sources

Sources sert à explorer le contenu des fichiers locaux ou distants utilisés par votre application. Ce visualiseur s'ouvre automatiquement lorsque vous cliquez sur un nom de fichier dans un autre onglet de Dev Tools.

Copyright © 2015 Eyrolles.

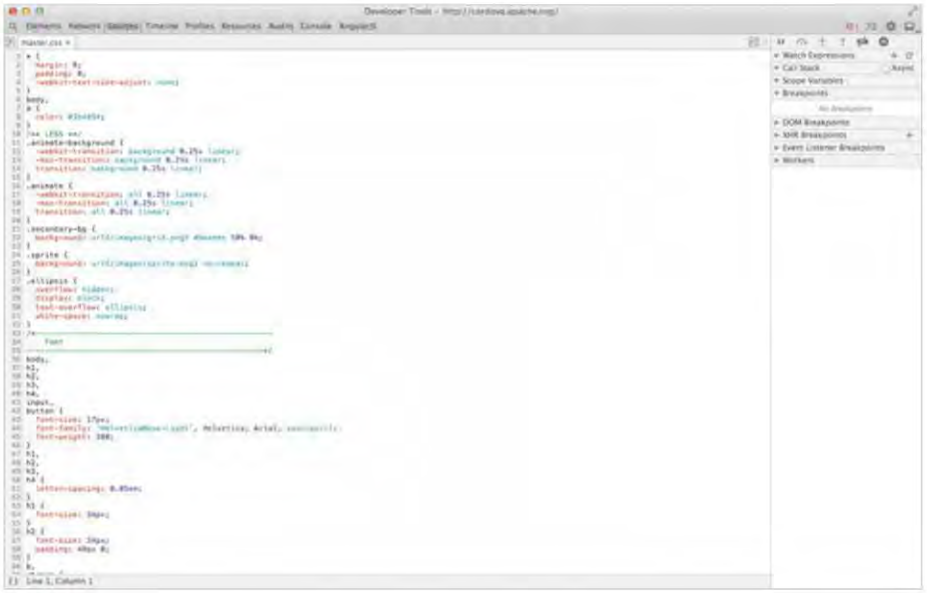


Figure 6-7
Onglet Sources de Google Chrome Dev Tools

Profiles

Profiles réalise un profilage de votre application, c'est-à-dire une analyse précise de divers aspects de l'application. Dans ce cas précis, vous pourrez étudier l'évolution de l'utilisation du CPU en rapport avec le moteur JavaScript, ou encore analyser l'évolution de la consommation mémoire utilisée par votre application, pour détecter une éventuelle fuite mémoire.

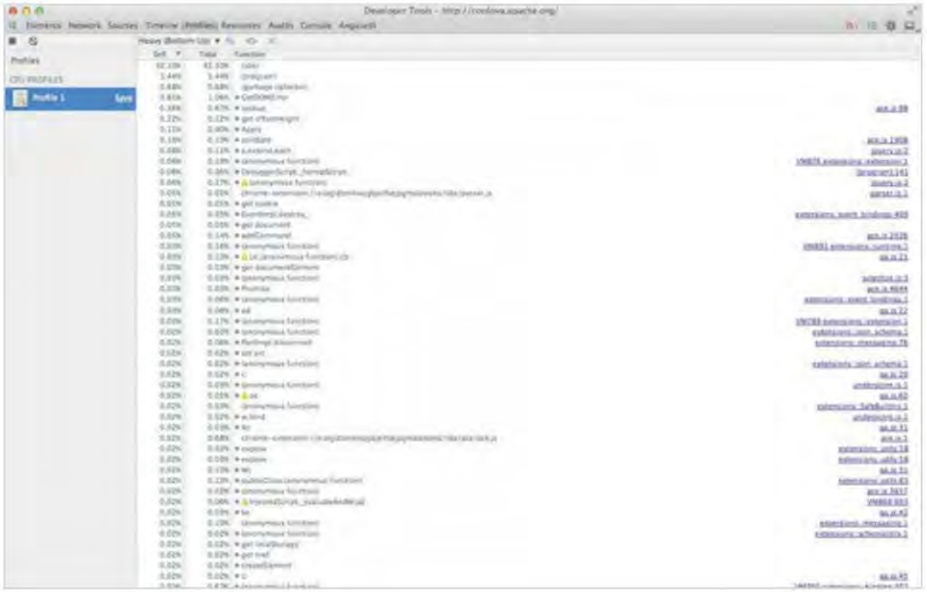


Figure 6-8
Onglet Profiles de Google Chrome Dev Tools

Copyright © 2015 Eyrolles.

Audits

Cet outil permet d'effectuer un audit de votre application et de recevoir en retour des conseils pour améliorer ses performances, comme activer la compression gzip, concaténer les fichiers CSS et JS, mettre en place un système de cache, etc. Notez que ces recommandations s'appliquent davantage à une WebApp car les fichiers d'une application Cordova sont généralement chargés depuis le système de fichiers local. Il n'est donc ainsi pas forcément utile de minifier/concaténer ces fichiers.

Figure 6-9
Onglet Audits de Google Chrome Dev Tools

Resources

Grâce à Resources, vous pouvez accéder à la gestion des données de votre application, et ainsi explorer le contenu de la base de données Web SQL, IndexedDB ou encore le Local Storage ou Session Storage. Vous serez également en mesure de modifier, d'ajouter ou de supprimer des valeurs pour réaliser vos tests.

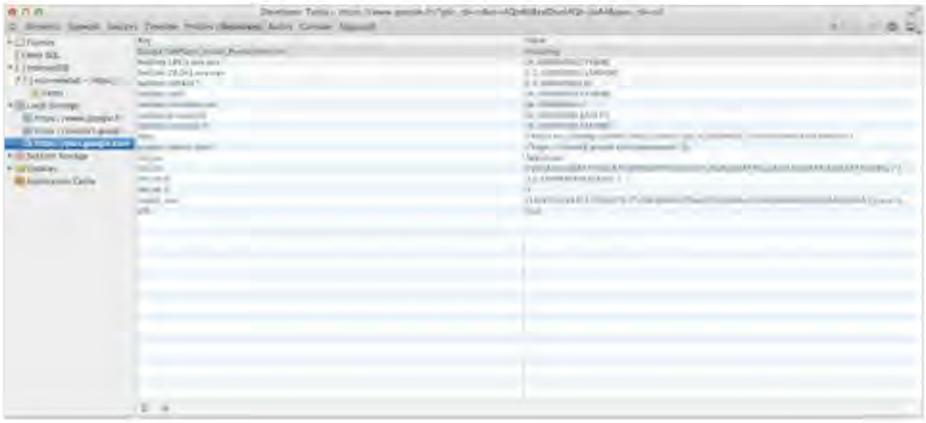


Figure 6-10
Onglet Resources de Google Chrome Dev Tools

Émuler un appareil mobile

Même si Chrome s'exécute sur une machine autre qu'un appareil mobile, on peut tout à fait simuler le fonctionnement d'un tel appareil, du moins dans une certaine mesure.

Comme nous l'avons vu, la fonction Emulation est disponible dans la seconde fenêtre de Chrome Dev Tools, accessible avec la touche Esc. Cette fenêtre comporte quatre onglets : Device, Screen, User Agent, Sensors.

L'onglet Device sert à choisir l'appareil à émuler. Il contient des références à une majorité des appareils mobiles du marché actuel : Amazon Kindle Fire, iPad, iPhone, BlackBerry, Google Nexus, HTC, LG, Motorola, Nokia, Samsung Galaxy ou encore Sony Xperia. Une fois l'appareil choisi, le bouton Emulate permet de démarrer l'émulation. Vous constaterez alors que la zone « active » de la fenêtre s'est réduite et que votre application, contenue dans cette zone, s'est donc rétrécie, selon les dimensions de l'écran simulé.

Copyright © 2015 Eyrolles.

Figure 6-11
Onglet Emulation/Device de Google Chrome Dev Tools

Le choix d'un appareil mobile dans cette liste est en fait un raccourci, c'est-à-dire qu'il constitue un préremplissage de valeurs dans les autres onglets. Par exemple, le choix de l'iPhone 5 permet de fixer les valeurs décrites ci-après :

```
Viewport: 640 x 1136, devicePixelRatio = 2
User agent: Mozilla/5.0 [...]
```

Nous verrons plus tard à quoi ces valeurs correspondent.

Pour quitter le mode Emulation, vous pouvez cliquer sur Reset ou simplement fermer la console.

Allez dans l'onglet Screen pour régler tout ce qui concerne la zone « active » de l'écran. Dans le cas de l'iPhone 5, cette zone est ainsi préremplie avec 640 pixels de large pour 1 136 pixels de haut. Un bouton d'échange entre les deux zones inverse les valeurs : particulièrement pratique pour passer du mode portrait au mode paysage. Le curseur, juste en dessous, permet de faire doucement varier la largeur et ainsi tester vos points de rupture, autrement dit les largeurs d'écran provoquant des changements dans la mise en page.

Figure 6-12
Onglet Emulation/Screen de Google Chrome Dev Tools

devicePixelRatio est le ratio entre les pixels physiques et les pixels logiques. Ainsi, un ratio de deux dans le cas de l'iPhone 5 veut dire qu'un pixel à l'écran représente en réalité deux pixels logiques : c'est ce qu'on appelle un écran Retina chez Apple. Si vous affichez dans votre application une image de 100 px par 100 px, il faudra que le fichier source soit d'au moins 200 px par 200 px pour avoir une résolution optimale. Une image en dessous de cette valeur apparaîtra floue sur un appareil Retina, même si elle apparaît correctement sur un appareil non Retina comme l'iPhone 3GS.

L'option Enable text autosizing simule le comportement des appareils mobiles concernant le traitement des tailles de polices.

L'option Emulate viewport fixe le zoom de la page sur la valeur du Viewport de l'appareil et non pas la valeur de la largeur physique.

L'option Shrink to fit fait en sorte que l'intégralité de l'appareil tienne dans la fenêtre du navigateur.

La section CSS media permet de simuler la présence de diverses feuilles de styles CSS, comme celles destinées à l'impression. En effet, il est possible de personnaliser le design d'une page à imprimer via un fichier CSS, pour par exemple masquer les menus, les publicités ou encore les aplats de couleur en général.

Copyright © 2015 Eyrolles.

L'onglet User Agent modifie le *User Agent* utilisé par l'appareil. Celui-ci est une chaîne de caractères qui est censée retranscrire la version de l'appareil et du navigateur qui sont utilisés, permettant dans certains cas de procéder à des adaptations particulières. Ainsi, un site pourrait appliquer des modifications uniquement aux appareils sous iOS ou Android via cette valeur. Toutefois, ceci n'est pas conseillé contrairement à la détection de fonctionnalités[1]. Les options de cet onglet permettent donc de choisir votre *User Agent*. L'utilisation du User Agent est toutefois largement remise en cause et déconseillée, car peu précise (et facilement falsifiable).

Figure 6-13
Onglet Emulation/User Agent de Google Chrome Dev Tools

L'onglet Sensors contient toutes les options qui concernent les différents capteurs de l'appareil, à savoir de l'écran, du GPS ou encore de l'accéléromètre. Vous pouvez ainsi émuler une position GPS (pratique pour effectuer vos tests de positionnement sans aller sur le terrain) ou encore des valeurs précises pour l'accéléromètre (utile pour éviter de manipuler un appareil manuellement avec toutes les imprécisions que cela entraîne, ou pour pallier l'absence d'accéléromètre sur les ordinateurs de bureau). L'option Emulate touch screen permet de transformer votre souris en un « doigt virtuel ». Vos clics sont alors détectés comme des événements `touchstart`, `touchmove`, `touchend`. Vous pouvez également maintenir la touche Shift enfoncée pour simuler des *pinch-in* ou *pinch-out*. Gardez en mémoire que l'émulation de `touch`, comme l'émulation de taille d'écrans ou beaucoup d'autres, ne fonctionne que si les Dev Tools sont ouverts. Dès que vous fermez la fenêtre, les paramétrages sont annulés.

Figure 6-14
Onglet Emulation/Sensors de Google Chrome Dev Tools

Dans un simulateur

Comme nous l'avons vu, pour des raisons de simplicité et de rapidité, une bonne partie du développement a lieu dans le navigateur. Mais il est très important de tester votre application de manière régulière dans un simulateur, puis sur un appareil mobile.

1. http://modernizr.com

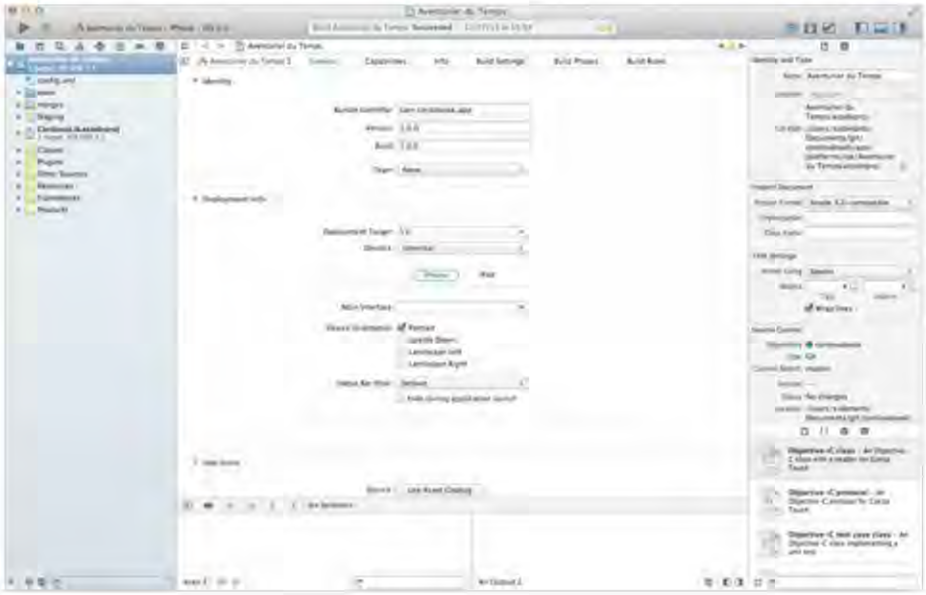
Copyright © 2015 Eyrolles.

Si vous développez une application pour iOS, le SDK dédié vous fournit tous les outils nécessaires pour tester dans un simulateur. Pour une application Android, des outils comme GenyMotion vous permettent de tester l'application sur diverses versions d'Android et appareils mobiles.

Debug dans un simulateur iOS

Lorsque votre application est prête à être testée dans un simulateur (autrement dit, dès le début !), utilisez la commande `cordova platform add ios`, puis `cordova build ios` pour créer le projet Xcode de votre application. Vous trouverez ce dernier dans le dossier `platforms/ios/VotreApp.xcodeproj` de votre projet Cordova. Double-cliquez sur ce fichier pour ouvrir le projet dans Xcode. Notez que vous pouvez également utiliser directement la commande `cordova emulate` et ainsi vous passer totalement d'Xcode.

Figure 6-15
Exemple de projet Xcode



Après le lancement d'Xcode, l'architecture de votre projet apparaîtra dans la partie gauche de l'écran. La partie qui nous intéresse se situe en haut à gauche de l'écran. Le bouton Play permet de lancer l'application sur l'appareil et la version d'iOS sélectionnées juste à côté.

Figure 6-16
Barre de lancement d'Xcode

Vous pouvez, par exemple, choisir parmi les appareils suivants : iPhone, iPhone Retina (3,5 pouces), iPhone Retina (4 pouces), iPhone Retina (4 pouces, 64-bit), iPad, iPad Retina et iPad Retina (64-bit).

Chacun de ces appareils est disponible dans les versions d'iOS que vous avez installées sur votre ordinateur. En effet, la dernière option du menu More Simulators permet d'installer les appareils et les versions d'iOS qu'il vous manque.

Copyright © 2015 Eyrolles.

Lorsque vous avez choisi votre appareil et votre version d'iOS, cliquez sur la touche Play pour lancer le simulateur. Ce dernier s'ouvre sous la forme d'une application dédiée appelée iOS Simulator. Votre application se lance alors dans un simulateur iPhone ou iPad, vous permettant d'interagir à l'aide de votre souris.

Figure 6-17
Simulateur iOS

Astuce

Votre souris remplace le doigt de l'utilisateur. Il faut donc effectuer un cliquer-déplacer pour scroller sur la page.

Le simulateur propose de nombreuses options intéressantes, accessibles par raccourcis clavier ou directement depuis le menu de l'application Simulateur. Voici les plus utiles…

- Fichier>Enregistrer la capture d'écran : sauvegarde une impression d'écran de la vue dans le simulateur (un fichier est alors créé sur votre bureau). Cette fonctionnalité est particulièrement utile pour créer les images à ajouter à iTunes Connect avant la soumission sur l'App Store.
- Matériel>Appareil : change l'appareil ou la version du simulateur actuellement utilisé.
- Matériel>Rotation à gauche/droite : simule une rotation de l'appareil, comme un utilisateur le ferait en retournant l'iPhone ou l'iPad entre ses mains.
- Matériel>Secousse : simule une secousse de l'appareil, permettant ainsi de tester d'éventuelles fonctionnalités que vous auriez codées en ce sens.
- Matériel>Écran d'accueil : accède au menu principal de l'iPhone ou iPad, ce qui s'avère utile pour désinstaller l'application, par exemple. Le raccourci Cmd + Shift + H répété deux fois de suite permet d'accéder à la barre de multitâches et ainsi de stopper l'application en la faisant glisser vers le haut de l'écran.
- Fenêtre>Échelle : modifie la taille de l'écran du simulateur. Dans le cas de l'iPhone classique, un zoom de 100 % vous permet, par exemple, de faire apparaître une « enveloppe » d'un réel iPhone autour de l'application, ce qui se révèle très utile pour effectuer des impressions d'écrans incluant l'appareil. Cette option sert également à adapter l'affichage du simulateur à la taille de l'écran de votre ordinateur.

Copyright © 2015 Eyrolles.

- Alt + souris : simule un pinch-in ou pinch-out avec deux doigts
- Alt + Shift + souris : simule un drag ou clic avec deux doigts.

Figure 6-18
Simulateur iOS penché vers la gauche

Pour le moment, nous avons vu comment exécuter votre application dans un simulateur, mais pas encore comment la débugger réellement. Pour cela, deux sources d'informations vous permettent de mieux comprendre le fonctionnement et les éventuels problèmes de votre application :

- la console d'Xcode ;
- l'outil de debug de Safari.

La console d'Xcode, située en bas de l'écran, affiche tous les messages d'information, d'avertissement ou d'erreur de votre application, tant en provenance du code natif Objective-C que du code JavaScript.

Figure 6-19
Console Xcode

Vous pourrez ainsi y découvrir la raison du dysfonctionnement d'un plug-in. Toutefois cet outil ne vous permet pas vraiment d'interagir avec l'application. Pour cela vous devez utiliser Safari.

Lorsqu'une application s'exécute dans un simulateur iOS, ou sur un appareil iOS relié à l'ordinateur, Safari est en mesure d'en explorer le code comme n'importe quel site Internet ouvert dans le navigateur. Pour cela, vous devez tout d'abord activer le menu Développeur de Safari, en vous rendant dans les paramètres, onglet Avancées, puis en cochant la case « Afficher le menu Développement dans la barre des menus ». Vous constaterez alors qu'une nouvelle entrée « Développement » a fait son apparition dans le menu de Safari. En dessous des deux premières options, vous découvrirez la liste des simulateurs et des appareils compatibles connectés à l'ordinateur; et pour chacune des entrées la liste des applications et sites Internet ouverts sur le simulateur ou l'appareil.

Figure 6-20
Menu Développement de Safari

Copyright © 2015 Eyrolles.

Astuce

Si votre appareil ou simulateur n'apparaît pas dans le menu, n'hésitez pas à redémarrer Safari ou à relancer l'application.

Sélectionnez la page `index.html` de votre application pour ouvrir l'outil de Développement de Safari. Les outils à disposition dans Safari ressemblent beaucoup à ceux proposés dans Google Chrome Dev Tools. L'explorateur de page qui s'ouvre par défaut permet, par exemple, de naviguer dans le DOM, et d'afficher en surbrillance certains éléments dans l'application. De même, de manière similaire à Chrome Dev Tools, la console permet d'interagir avec l'application en exécutant du code JavaScript.

Figure 6-21
Outils de développement Safari

Astuce

L'ouverture des outils de debug Safari ne peut se faire qu'après le lancement de l'application dans le navigateur. Vous pourriez en effet rater des messages ou d'éventuelles erreurs survenant lors de la phase de chargement. Afin d'éviter ce problème, placez une ligne `alert("stop")` au tout début de votre code JavaScript et dans la fonction `onDeviceReady` pour ainsi bloquer temporairement l'exécution du reste du code. Cela vous laissera le temps d'ouvrir la console de Safari avant de reprendre l'exécution en cliquant simplement sur OK dans la boite de dialogue.

Debug dans un simulateur Android

Contrairement à l'environnement iOS, où tous les outils sont intégrés, le développement Android nécessite, pour davantage de confort, l'installation d'outils externes, toutefois optionnels. Dans notre cas, nous allons avoir besoin des outils GenyMotion et weinre, mais surtout du plug-in ADT.

ADT

ADT *(Android Development Tools)* est un plug-in pour Eclipse, l'éditeur le plus utilisé pour construire et gérer des projets de développement notamment en Java.

Copyright © 2015 Eyrolles.

Si vous aviez choisi de n'installer que le SDK « allégé » en partie 1 (voir chapitre 2), vous devrez ici télécharger la dernière version d'Eclipse IDE for Java Developers en vous rendant sur le site officiel http://www.eclipse.org/downloads/. Cet ouvrage ne couvre pas en détail l'installation et la configuration d'Eclipse, car vous trouverez de nombreux tutoriels en ligne sur le sujet.

Une fois votre instance d'Eclipse installée, vous pouvez vous procurer la dernière version d'ADT. La procédure complète est détaillée sur le site officiel[2].

Astuce

Pour les installations de ces environnements, essayez de vous référer au maximum aux documentations officielles. En effet, celles-ci sont généralement à jour, tandis que les nombreux forums de discussion portent généralement sur des versions plus anciennes des suites logicielles.

Vous êtes maintenant prêt à créer votre premier projet Android ! Comme vous l'avez peut-être remarqué pour iOS, la plupart du travail s'effectue dans Cordova, et non dans les outils propriétaires. ADT ne déroge pas à la règle, puisque c'est bien Cordova qui va créer le projet Android pour vous.

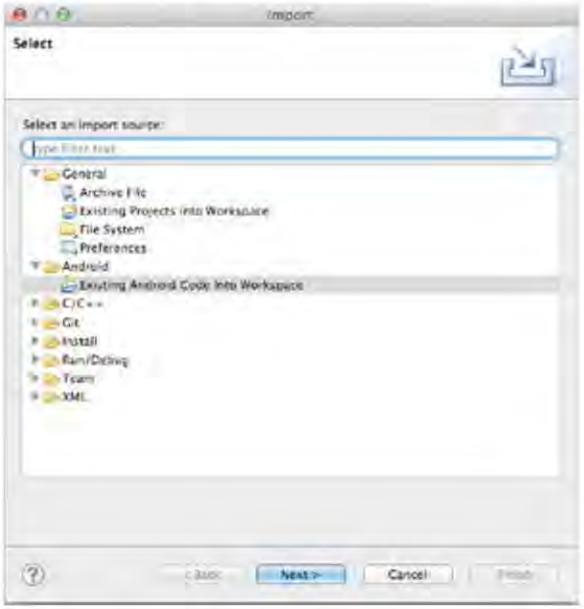
En effet, lorsque vous aurez une première version à tester sur Android, utilisez la ligne de commande pour vous rendre dans le dossier de votre projet Cordova, et exécutez la commande `cordova build android`.

Rappel

Pour rappel, vous deviez au préalable ajouter Android comme plate-forme cible avec la commande `cordova platform add android`.

La commande `build` crée pour vous le projet Android dans le dossier `platforms/android`. Après l'exécution avec succès de la commande, rendez-vous dans Eclipse ADT, puis dans File>Import.

Figure 6-22
Import d'un projet dans ADT



2. http://developer.android.com/sdk/installing/installing-adt.html

Copyright © 2015 Eyrolles.

Une fenêtre s'ouvre alors, vous demandant de choisir le type de projet que vous voulez importer. Choisissez Existing Android Code Into Workspace.

Sur l'écran suivant, naviguez jusqu'au dossier `platforms/android` de votre projet Cordova, et validez. ADT va alors détecter deux projets : celui de votre application, et celui de la bibliothèque CordovaLib. Vérifiez que ces deux projets sont sélectionnés, et cliquez sur Finish.

Figure 6-23
Exemple d'import d'un projet dans ADT

Votre projet apparaît maintenant à gauche dans la liste des projets. Il est prêt à être exécuté sur un simulateur grâce à GenyMotion !

GenyMotion

GenyMotion se présente comme l'émulateur Android le plus rapide au monde. Avec 900 000 utilisateurs à travers la planète, il constitue un équivalent sérieux aux simulateurs inclus dans Xcode pour iOS.

GenyMotion se base sur VirtualBox, un logiciel permettant d'exécuter des machines virtuelles. Une machine virtuelle est l'équivalent « virtuel » d'une machine physique. Autrement dit, une même machine physique peut embarquer plusieurs machines virtuelles. Ainsi, votre ordinateur peut émuler plusieurs versions d'Android, une image d'un Windows 7 ou encore d'un Windows Server 200, le tout sur la même machine à une portée de clic.

Après avoir installé la dernière version de VirtualBox, disponible en ligne (https://www.virtualbox.org/wiki/Downloads), vous pouvez installer GenyMotion directement depuis leur site Internet (https://cloud.genymotion.com/page/launchpad/download/).

Au démarrage de GenyMotion, vous devez procéder à l'étape d'installation des différents terminaux virtuels dont vous avez besoin.

Copyright © 2015 Eyrolles.

Figure 6-24
Page d'accueil de GenyMotion avec plusieurs terminaux

Cliquez sur le bouton Ajouter pour accéder à la liste des appareils disponibles. Si vous savez ce que vous recherchez, utilisez le champ de recherche. Vous pouvez également dérouler les listes pour sélectionner une version d'Android ou un modèle de téléphone. Ensuite, suivez simplement le processus d'installation.

Astuce

Pensez à installer des appareils de version Android et de résolutions différentes pour que vos tests couvrent suffisamment de cas possibles.

Une fois vos terminaux installés, vous allez pouvoir démarrer votre premier simulateur. Sélectionnez l'appareil de votre choix, et cliquez sur Play. VirtualBox se lance de manière transparente, et un téléphone Android apparaît. À la manière des simulateurs iOS, vous pouvez alors utiliser le terminal virtuel comme un vrai téléphone, sans oublier que le doigt de l'utilisateur est remplacé par votre souris. Servez-vous du cliquer-déplacer pour effectuer des actions comme le déverrouillage du téléphone par exemple !

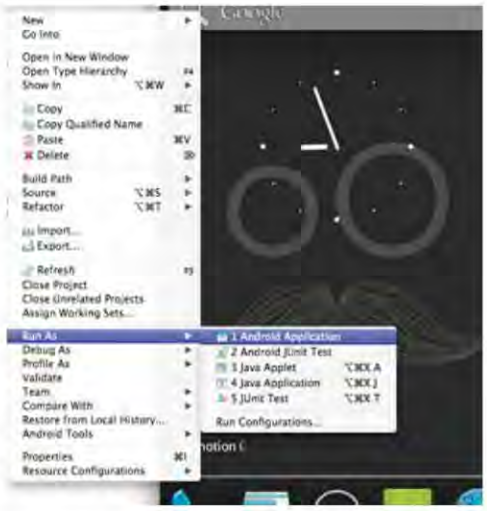
Figure 6-25
Simulateur GenyMotion

Copyright © 2015 Eyrolles.

GenyMotion propose un certain nombre d'outils pour vous aider au debug. Par exemple, les trois premières icônes en haut à droite permettent de régler le niveau de batterie, le GPS ou encore de prendre une photo. Vous trouverez également parmi les boutons à droite un bouton de retour permettant de tester le bon fonctionnement de votre code de détection du bouton Retour physique.

Une fois que votre simulateur est lancé, retournez dans ADT. Effectuez un clic droit sur votre projet, et cliquez sur Run As>Android Application.

Figure 6-26
Exécuter une application Android



La compilation de votre application va alors débuter. Elle se termine par l'affichage d'une fenêtre présentant les différents appareils Android disponibles pour la simulation. Votre simulateur GenyMotion devrait apparaître dans la liste. Sélectionnez-le et cliquez sur OK.

Figure 6-27
Choix d'un simulateur Android

Bravo, votre application s'exécute maintenant dans un simulateur Android !

Copyright © 2015 Eyrolles.

weinre

Même si Chrome permet de débugger une application lancée sur un appareil mobile, cette astuce ne fonctionne pas avec GenyMotion. Il faut donc utiliser un autre outil : weinre. Prononcé *winery* en anglais, il s'agit d'un inspecteur web conçu pour fonctionner sur des pages à distance.

Commencez par installer weinre sur votre machine :

```
sudo npm -g install weinre
```

Récupérez ensuite votre adresse IP locale à l'aide de la commande :

```
ifconfig
```

Lancez le serveur weinre sur votre machine en précisant votre adresse IP après le paramètre `boundHost` :

```
weinre --boundHost X.X.X.X --httpPort 8080
```

Ajoutez la ligne suivante dans le code votre application, juste avant la balise `</body>` du fichier `index.html` par exemple, en précisant l'adresse IP de votre serveur.

```
<script src="http://X.X.X.X:8080/target/target-script-min.js#anonymous"></script>
```

Vous pouvez remplacer `anonymous` par le nom de votre choix.

Ouvrez ensuite la page suivante dans votre navigateur Chrome, en prenant soin d'utiliser votre propre adresse IP, et le cas échéant de remplacer `anonymous` par la valeur choisie à l'étape précédente :

```
http://X.X.X.X:8080/client/#anonymous
```

Vous devriez vous retrouver devant l'écran de la figure 6-28.

Figure 6-28
Accueil de weinre

Puis compilez votre application et lancez-la dans un simulateur GenyMotion. Au démarrage de l'application, si vous n'avez raté aucune étape, vous devriez voir apparaître votre appareil dans la section Target. Si celle-ci n'est pas sélectionnée, cliquez sur l'entrée correspondante dans la liste. Vous aurez ensuite accès à des outils similaires à ce que Chrome Dev Tools propose, à savoir une console, un explorateur de DOM, un analyseur de trames réseau et un accès aux ressources.

Copyright © 2015 Eyrolles.

Figure 6-29
Target dans weinre

Gardez à l'esprit que weinre peut parfois se révéler assez instable, avec un arbre DOM qui ne se met pas à jour, ou encore des styles CSS qui n'apparaissent pas dans l'inspecteur de styles. Vous devrez peut-être relancer la page weinre ou votre application dans certains cas. Mais cet outil peut être très utile, notamment si un bug apparaît sur un appareil mobile particulier. Notez également que rien n'empêche a priori weinre de fonctionner à distance si votre serveur possède une IP publique. Vous pourriez donc envoyer l'application à toute personne en charge des tests, et depuis votre ordinateur, surveiller sa connexion sur le serveur weinre.

Sur un appareil

Nous l'avons vu précédemment, les tests dans un navigateur ou même sur un simulateur ne remplaceront jamais ceux effectués sur un véritable appareil mobile. En effet, les tests opérés en situation réelle peuvent mettre en évidence les points suivants :

- présence de zones difficilement cliquables ;
- écran blanc au démarrage avant l'apparition de la première vue ;
- transition peu esthétique entre les vues, due à la lenteur du téléphone ;
- latence sur les scrolls ;
- erreurs si le téléphone n'est pas connecté à Internet.

Mais le principal atout de ces tests est de vous apporter un ressenti de votre application, similaire à ce que connaîtront vos futurs utilisateurs, ce qui est absolument primordial.

Debug iOS

Le debug d'une application iOS exécutée sur un appareil mobile est relativement simple. Nous savons déjà que Safari permet de débugger le code d'une application exécutée dans le simulateur Xcode ; Safari peut de la même manière servir à débugger une application exécutée dans un appareil mobile iOS connecté à votre ordinateur via un câble USB.

Pour cela, branchez simplement votre appareil mobile ouvrez Safari sur votre ordinateur et lancez l'application depuis Xcode sur votre appareil mobile. Une fois que le démarrage de celle-ci est terminé, vous pourrez voir le nom de votre iPhone ou iPad en haut du menu Développement de Safari. Le sous-menu à droite du nom de votre appareil laissera alors apparaître le nom de votre application (accompagné de sa page d'accueil, a priori `index.html`) et des éventuels sites

Copyright © 2015 Eyrolles.

web ouverts dans Safari sur votre appareil. Sélectionnez la page d'accueil de votre application pour ouvrir les outils de debug de Safari. Vous voilà prêt à explorer le DOM, consulter la console d'erreur ou encore injecter du code JavaScript !

Debug Android ou iOS avec weinre

Nous vous présentions précédemment le système weinre qui permet de débugger une page ou une application ouverte sur un simulateur. Sachez que ce système fonctionne également pour les applications exécutées sur des appareils mobiles, tant Android qu'iOS.

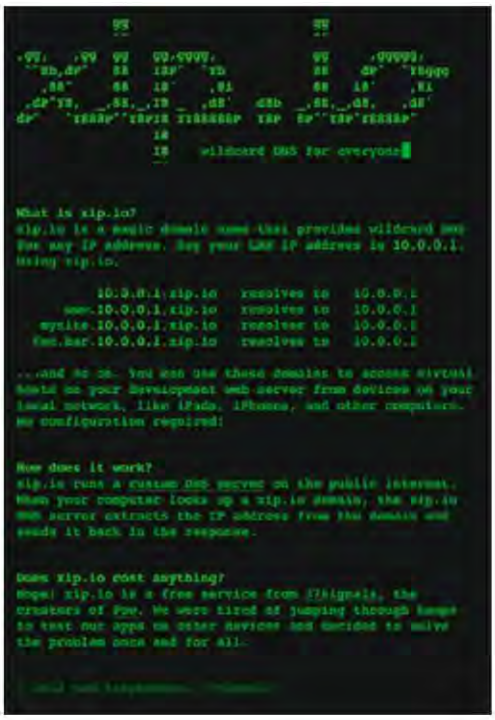
Accéder à un serveur local

Si vous hébergez sur votre machine l'API à laquelle votre application fait appel, vous allez rapidement rencontrer un problème. En effet, si votre API est hébergée derrière un VirtualHost (par exemple http://api.monapp.com), vous allez vous rendre compte que votre application lancée dans un appareil mobile ne peut accéder à cette API, même en étant sur le même réseau Wi-Fi. La raison est simple : le fichier `hosts` enregistré sur votre ordinateur n'est pas vu par les appareils mobiles de votre réseau. Et le fichier `hosts` d'un iPhone, par exemple, n'est pas paramétrable.

La première solution qui vient à l'esprit est donc de mettre en production (ou en préproduction) votre API pour pouvoir tester l'application sur vos appareils mobiles. Mais cela nécessite de disposer d'un serveur de préproduction ou de production, ce qui n'est pas toujours le cas au début du développement.

Fort heureusement, il existe une seconde solution plus flexible : xip.io.

Figure 6-30
Page d'accueil de xip.io



Copyright © 2015 Eyrolles.

xip.io est un service permettant de résoudre une adresse publique en une adresse locale. Par exemple, l'adresse `api.192.168.0.1.xip.io` va être résolue par le DNS de xip.io pour pointer sur votre API locale hébergée sur votre serveur à l'adresse `192.168.0.1`. Votre appareil mobile pensera donc dialoguer avec un serveur en ligne, alors qu'il sera redirigé vers votre serveur local, qui acceptera de lui répondre.

L'utilisation de xip.io est très simple. Ajoutez la ligne suivante à votre VirtualHost :

```
ServerAlias monapi.*.xip.io
```

Redémarrez votre serveur local. Remplacez dans votre application l'adresse de votre API par l'adresse suivante, en remplaçant `192.168.0.1` par l'IP de votre serveur sur le réseau local :

```
http://monapi.192.168.0.1.xip.io/
```

Astuce

Attention, gardez à l'esprit que le service xip.io ne fonctionne pas avec toutes les box personnelles. La Freebox Révolution, par exemple, ne permet pas l'utilisation de ce service. Vous pouvez alors vous tourner vers le service ngrok, plus complet que xip.io, mais légèrement plus compliqué à configurer.

Répercuter les changements sans recompiler

Dans le monde du développement web classique, il existe de nombreux outils surveillant les changements apportés aux fichiers CSS, JavaScript et HTML pour ensuite donner l'ordre au navigateur Internet de recharger la page sans action manuelle supplémentaire. Ainsi, chaque modification est appliquée en temps réel et le processus de debug n'en devient que plus confortable. LiveReload[3], CodeKit[4] et Prepros[5] font partie des outils les plus connus ; certains sont gratuits, tandis que d'autres, payants, intègrent toute une batterie de fonctionnalités complémentaires fort intéressantes telles que l'optimisation des images ou encore la minification/concaténation des fichiers JavaScript et CSS.

En raison de l'étape de compilation requise pour le déploiement des applications hybrides dans un simulateur ou sur un appareil mobile, les équivalents à LiveReload pour Cordova ont tardé à voir le jour. En effet, ce n'est qu'avec l'arrivée de Cordova 3 que de tels outils ont pu être mis en place, par le biais de l'interface en lignes de commandes. Ainsi, il est aujourd'hui possible de déployer une application simultanément dans différents simulateurs et appareils mobiles, de développer dans l'éditeur de son choix, et de voir tout changement être automatiquement répercuté sans ne jamais avoir à recompiler quoi que ce soit ; le tout en débuggant à distance depuis Google Chrome ou Safari. Un vrai bonheur !

3. http://livereload.com/
4. https://incident57.com/codekit/
5. https://prepros.io/

Copyright © 2015 Eyrolles.

Encore une fois, plusieurs solutions existent ici[6], chacune ayant ses avantages et inconvénients. Adobe fournit même une application dédiée[7] globalement très simple d'utilisation, mais celle-ci ne prend actuellement pas en charge l'ajout de plug-ins tiers. Bien avant son arrivée, nous avions d'ailleurs développé GapReload[8], un plug-in open source basé sur LiveReload, configurable à souhait et accompagné d'une tache Grunt[9] pour davantage de flexibilité. GapReload prend quant à lui en charge les plug-ins tiers. Vous n'avez que l'embarras du choix, et c'est tant mieux !

6. http://developer.telerik.com/featured/bringing-f5-or-commandr-to-hybrid-mobile-apps/
7. http://app.phonegap.com/
8. http://plugins.cordova.io/#/package/pro.fing.cordova.gapreload
9. https://github.com/fingerproof/grunt-gapreload

Copyright © 2015 Eyrolles.

Partie III

Diffusion d'une application

De la même manière qu'un site web doit être mis en ligne pour être disponible à la vue de tous, une application mobile doit être mise à disposition sur des magasins d'applications. L'App Store est le magasin des applications iOS et le Google Play Store celui d'Android. Tous deux ont un fonctionnement relativement proche : création de la page de votre application (titre, description, logo, etc.) et soumission du code source de votre application. Très pratiques pour les utilisateurs, les magasins d'applications présentent cependant un inconvénient pour les développeurs, car votre application sera noyée dans la masse. Il convient donc de soigner vos mots-clés, votre description, vos captures d'écran, et d'obtenir de nombreux téléchargements de votre application pour faire remonter cette dernière dans les résultats de recherche et ainsi dans les classements !

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

7

Les magasins d'applications

Publication sur l'App Store

La publication de votre application sur l'App Store d'Apple est un processus relativement complexe, qu'il faut aborder calmement pour ne rater aucune étape. Notez toutefois qu'avec le temps, Apple améliore ses outils, lesquels deviennent de plus en plus intuitifs. Et avec l'habitude, vous apprendrez à les utiliser sans même y penser.

Deux outils vont vous être indispensables pour publier votre première application : Member Center (iOS Dev Center[1]) et iTunes Connect[2].

Member Center permet de gérer tous les aspects techniques de la publication, à savoir les certificats qui attestent de la provenance de votre application ou encore la liste des appareils autorisés à tester votre application.

iTunes Connect sert à créer l'application et à gérer sa présentation (images d'illustration, description, choix des pays où publier…).

Enfin, nous aborderons également l'outil TestFlight qui permet de faire tester facilement votre application à d'autres utilisateurs iOS.

1. https://developer.apple.com/devcenter/ios/index.action
2. https://itunesconnect.apple.com/WebObjects/iTunesConnect.woa

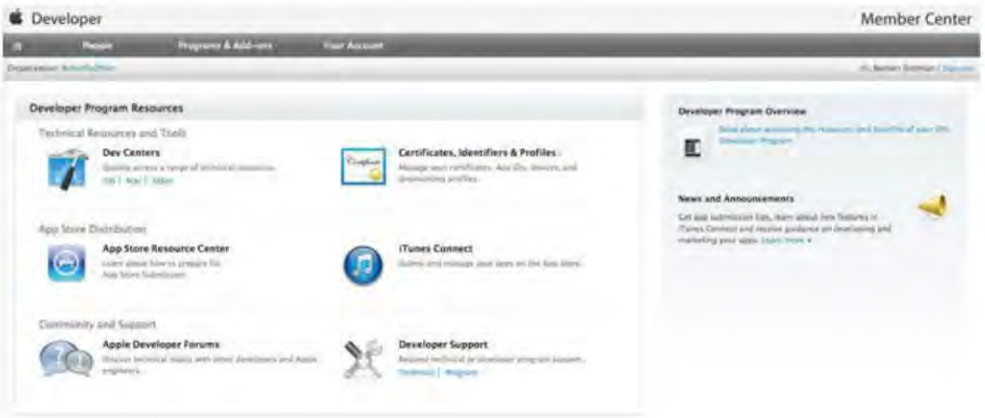
Copyright © 2015 Eyrolles.

Gestion des certificats sur Member Center

Member Center est un espace réservé sur le site d'Apple, accessible avec votre mot de passe développeur à l'adresse suivante https://developer.apple.com/membercenter/. Il permet principalement de gérer :

- les identifiants de vos applications ;
- les certificats de développeur ;
- la liste des appareils liés à votre application ;
- les profils de signature de votre application.

Figure 7-1
Accueil du Member Center



Une fois connecté au Member Center, cliquez sur la rubrique en haut à droite intitulée « Certificates, Identifiers & Profiles ».

Figure 7-2
Accueil du Dev Center

Copyright © 2015 Eyrolles.

L'écran suivant permet d'accéder aux quatre rubriques principales présentées précédemment, dans le cadre de votre abonnement Développeur iOS : iOS App IDs, Certificates, Devices, Provisioning profile.

Note

Nous n'avons pas détaillé cette étape, mais pour publier une application sur l'App Store vous devez préalablement vous acquitter de l'abonnement Développeur iOS d'une valeur de 99 dollars américains par an. Cet abonnement doit être renouvelé tous les ans pour que vos applications restent disponibles au téléchargement. Il existe également un compte Entreprise offrant plus de possibilités, notamment la distribution d'applications internes à destination de vos employés et ne nécessitant pas de passer par le processus de validation Apple.

iOS App IDs

Commençons par créer l'identifiant unique de votre application. Pour cela, cliquez sur Identifiers. La création de l'identifiant est simple, choisissez une description (par exemple : « Rappelle-toi »), puis un `Bundle ID` (par exemple : `com.cordobook.app`). Après validation, votre application dispose d'un identifiant unique auprès d'Apple.

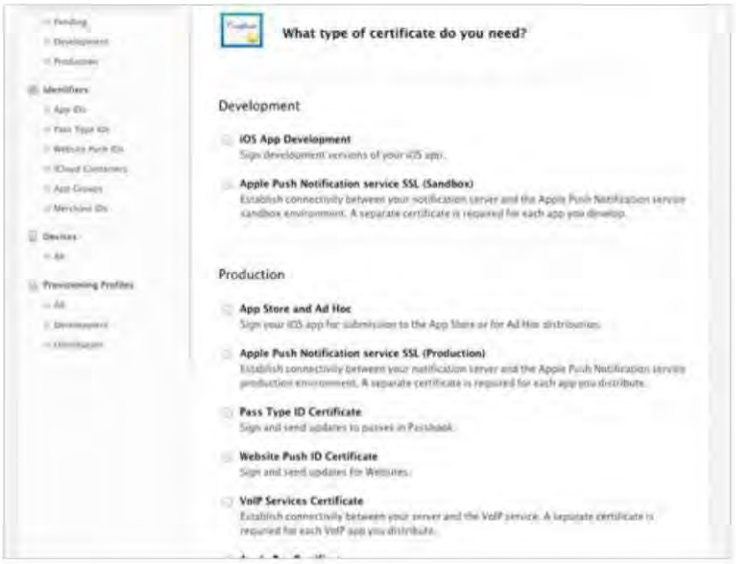
Figure 7-3
Enregistrer un identifiant unique d'application

Certificates

Créons maintenant le certificat permettant de signer votre application. Vous devez en créer deux : un pour le développement et un pour la production. Cliquez sur la rubrique Certificates>All à gauche de votre écran, puis sur le symbole « + ».

Copyright © 2015 Eyrolles.

Figure 7-4
Ajout d'un certificat

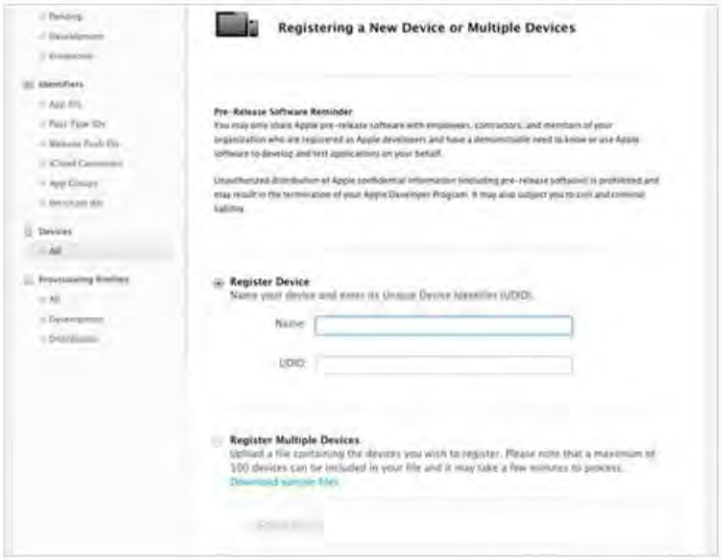


Choisissez dans un premier temps la première option iOS App Development et suivez le guide. Réitérez ensuite le même processus en choisissant App Store and Ad Hoc dans la section Production.

Devices

Rendez-vous maintenant dans la section Devices à gauche. Nous allons y enregistrer les appareils qui sont autorisés à exécuter l'application pendant la période de test. Cela inclut tous vos appareils iOS, ainsi que ceux de vos éventuels testeurs. Cliquez sur le bouton « + » en haut à droite pour commencer le processus.

Figure 7-5
Ajout d'un appareil



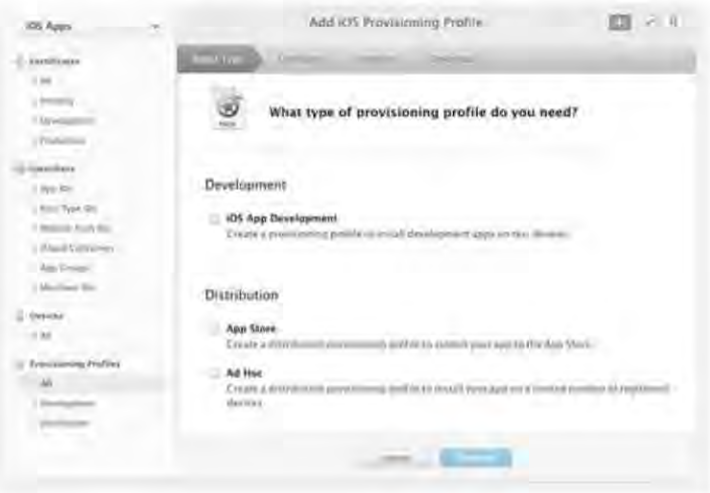
Copyright © 2015 Eyrolles.

Un appareil est composé d'un couple Name et UDID. Dans le champ Name, décrivez avec précision l'appareil (par exemple : « iPhone 6+ Sébastien P. »). Le champ UDID contient l'identifiant unique de l'appareil. Il existe diverses manières de le récupérer, comme des sites Internet qui permettent de le recevoir par e-mail, ou encore en utilisant iTunes pour l'afficher[3]. Validez le formulaire, et recommencez pour tous les autres appareils.

Provisioning profile

Rendez-vous maintenant dans la section Provisioning Profile à gauche. Un profil de provisionnement permet de relier un compte développeur, une application et un appareil mobile. Nous allons créer trois profils de provisionnement pour notre application : un profil de développement, un profil de production, et un profil ad hoc, dont nous verrons l'utilité un peu plus loin. Choisissez tour à tour chacune des trois options, et créez vos profils. Pour les profils de développement et ad hoc, il est primordial de sélectionner vos appareils de test à l'étape de configuration.

Figure 7-6
Ajout d'un profil



Astuce

Depuis peu, Xcode permet de réaliser certaines de ces actions directement depuis l'interface du logiciel. Nous avons cependant privilégié l'étape manuelle dans Member Center, qui donne un meilleur aperçu des différentes composantes nécessaires.

Publication via iTunes Connect

Le processus de publication de votre application sur l'App Store se compose de deux étapes : la création de l'application sur le portail iTunes Connect, et l'envoi du code de votre application.

3. http://whatsmyudid.com

Copyright © 2015 Eyrolles.

Création de votre application

Connectez-vous sur le portail iTunes Connect (https://itunesconnect.apple.com) à l'aide de vos identifiants Apple.

Figure 7-7
Accueil d'iTunes Connect

Le tableau de bord d'iTunes Connect vous permet d'accéder à la liste de vos applications, à des statistiques bien utiles, ou encore à l'historique de vos paiements. Commencez par visiter la section Mes apps et cliquez sur le bouton « + » en haut à gauche.

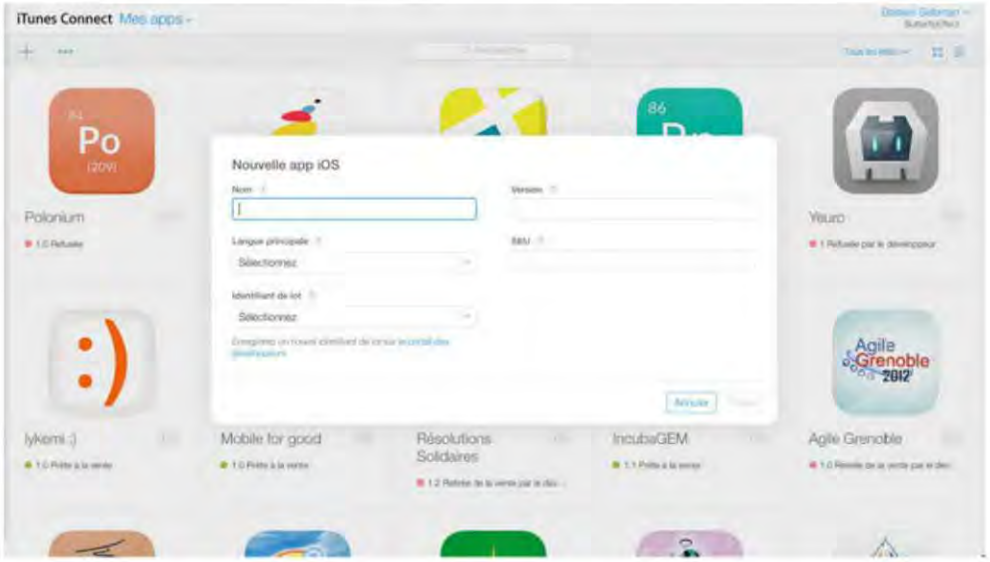


Figure 7-8
Mes apps sur iTunes Connect

Vous allez devoir remplir les premières informations concernant votre application :

Copyright © 2015 Eyrolles.

- Nom : le nom de l'application tel qu'il apparaîtra sur l'App Store ;
- Langue principale : la langue principale de l'application ;
- Identifiant de lot : l'identifiant de votre application défini dans Xcode, au format `com.cordobook.app` ;
- Version : le numéro de la version de votre application, par exemple « 1.0 » ;
- SKU : un numéro d'identification unique, par exemple « 1 ».

Après validation, vous serez redirigé vers l'interface de configuration avancée, comprenant quatre sections principales :

- Informations sur la version : définition des écrans de chargement, nom, description… ;
- Informations générales sur l'app : icône, version, catégorie, contact ;
- Informations utiles à la vérification de l'app : coordonnées du développeur, compte de test ;
- Publication de la version : réglage concernant la publication après revue.

La première étape consiste à mettre en ligne les images ou vidéos qui vont illustrer votre application sur l'App Store. Ces visuels sont à ajouter dans plusieurs formats, correspondants aux différents formats des appareils Apple.

Figure 7-9
Visuels de l'application sur iTunes Connect

Astuce

On ne le sait pas forcément, mais ces visuels n'ont pas besoin de correspondre aux écrans de chargement et captures d'écran de l'application. Ils peuvent également représenter un appareil affichant l'application, agrémentés de commentaires mettant en avant les fonctionnalités.

La deuxième étape consiste à écrire la description de l'application sur l'App Store. Celle-ci est très importante, car elle constitue, avec les visuels précédemment choisis, le premier contact des utilisateurs avec votre application. C'est également l'occasion de choisir les mots-clés qui décrivent le mieux votre application, et vous permettront de ressortir dans les résultats de recherche.

Copyright © 2015 Eyrolles.

Figure 7-10
Description de l'application sur iTunes Connect

C'est à l'étape suivante qu'on définit les informations générales de l'application, comme l'icône, la catégorie et l'identité de l'auteur.

Figure 7-11
Informations générales de l'application sur iTunes Connect

Enfin, la dernière étape consiste à préciser les coordonnées de la personne à contacter en cas de besoin lors de la revue de l'application. Les champs « compte démo » servent quant à eux à spécifier l'identifiant/mot de passe qu'Apple peut utiliser pour tester votre application.

Figure 7-12
Informations de publication de l'application sur iTunes Connect

Copyright © 2015 Eyrolles.

Astuce

Comme vous connaissez l'identifiant que va utiliser le testeur de l'équipe Apple, vous êtes en mesure de surveiller à quel moment il en est en train de tester, et même de voir en temps réel ce qu'il effectue dans l'application si vous utilisez un outil comme Google Analytics.

La dernière section permet de choisir si l'application sera publiée automatiquement après une revue favorable, ou seulement lorsque vous l'aurez décidé.

Envoi du code de l'application

Lorsque votre application est prête, qu'elle a été testée sur divers appareils virtuels et réels, que le profil de l'application sur iTunes Connect est complet, et que vous avez créé un profil de provisionnement de production, vous pouvez soumettre l'application à Apple. Pour cela, cliquez sur le bouton Soumettre à vérification sur la page du profil de l'application sur le portail iTunes Connect et suivez le guide.

Une fois qu'iTunes Connect indique être en attente du code de votre application, vous aurez le choix entre deux solutions pour uploader le fichier IPA requis.

La première, très simple, consiste à se rendre dans la section Ressources et aide d'iTunes Connect via le menu déroulant situé tout en haut à gauche de la page, puis à télécharger l'Application Loader 3.0 en cliquant sur le lien dédié sous le titre Préparation et livraison des apps. Vous trouverez également au même endroit l'Application Loader User Guide, soit la notice d'utilisation du programme, disponible en français. Une fois l'installation terminée, il vous suffit alors de vous connecter avec vos identifiants de développeur Apple, de cliquer sur Distribuer votre app, de choisir le fichier IPA à uploader dans l'arborescence de votre ordinateur, et enfin de suivre les quelques indications affichées à l'écran jusqu'à confirmation du bon déroulement de la procédure. C'est tout, et c'est aussi la voie conseillée pour toute application compilée via PhoneGap Build.

En second choix, rendez-vous dans Xcode et ouvrez le projet de votre application. Dans le menu de sélection de l'appareil en haut à gauche, choisissez la première entrée (iOS Device ou le nom de votre appareil si celui-ci est branché). Cette étape est primordiale si vous souhaitez publier votre application.

Cliquez ensuite sur Product>Archive. Xcode va alors préparer une archive de votre application et ouvrir la fenêtre Organizer qui contient toutes vos archives.

Figure 7-13
Exemple de fenêtre de l'Organizer dans Xcode

Avant de soumettre l'application à Apple, nous vous conseillons de passer par l'étape de validation qui vérifie que votre application est prête. Pour cela, cliquez sur Validate et suivez le processus en sélectionnant éventuellement votre compte développeur puis le profil de provisionnement de production de votre application.

Copyright © 2015 Eyrolles.

Astuce

À cette étape, si Xcode ne trouve pas votre *provisioning profile* de production, vérifiez que celui-ci a bien été créé en ligne. En outre, contrôlez aussi que le champ bundle identifier dans Xcode correspond bien à celui choisi lors de la création de l'App ID sur Member Center. Si votre profil n'apparaît toujours pas dans la liste, rendez-vous dans Xcode>Preferences>Account, sélectionnez votre compte, et cliquez sur View details. Vous devriez être en mesure de rafraîchir la liste et voir tous vos profils disponibles.

Si la validation se termine sans erreur, vous pourrez cliquer sur Submit et suivre le guide. Après d'ultimes réglages, Xcode soumettra votre code à Apple et vous préviendra du bon déroulement de la procédure. Dans le cas contraire, corrigez les erreurs, recréez une archive et recommencez le processus.

Astuce

Si la barre de chargement de l'envoi n'avance définitivement pas, c'est peut-être parce que votre réseau informatique ne permet pas l'utilisation du port employé par Xcode pour l'envoi du code. Dans ce cas, essayez de vous connecter à un autre réseau, ou encore au travers de la connexion Internet de votre téléphone si par chance vous disposez d'un forfait données 4G.

Si vous êtes arrivé au bout de l'étape d'envoi, félicitations, votre application a été reçue par Apple qui va maintenant la passer en revue !

Revue de votre application par Apple

Lorsque votre application aura été soumise avec succès auprès d'Apple, l'attente peut commencer. Apple met généralement une semaine pour tester l'application. En réalité, le test de votre application ne dure généralement que quelques minutes ! Sachez que les mises à jour futures de votre application seront également soumises à cette revue.

Astuce

Si pour une « bonne » raison vous avez besoin que la revue Apple soit plus rapide (l'application doit être disponible pour un événement particulier), vous pouvez exceptionnellement demander une revue expéditive. Mais ne tentez pas de jouer cette carte trop souvent, car cela pourrait s'avérer vite pénalisant. Pour en faire la demande, rendez-vous sur :

http://developer.apple.com/appstore/contact/?topic=expedite

Les personnes qui testent votre application vont vérifier de nombreux critères, qui sont listés dans l'App Store Review Guidelines (https://developer.apple.com/app-store/review/guidelines/). Nous vous invitons à lire ce document dans son ensemble pour avoir bien en tête ce qu'il est possible de faire ou non. À noter qu'Apple ne limite pas ses tests aux fonctionnalités, mais vérifie également le design de l'application, ainsi que son ergonomie. Un design de mauvaise qualité vous vaudra à coup sûr un refus.

Après une semaine d'attente, s'il s'avère qu'Apple refuse votre application, pas de panique ! Par expérience, nous pouvons vous dire qu'il arrive souvent qu'Apple refuse des applications, pas

Copyright © 2015 Eyrolles.

toujours pour de bonnes raisons. Les refus sont toujours motivés par une explication très précise des points bloquants, qui peuvent être généralement de trois natures :

- l'application ne respecte pas les Guidelines. Par exemple, vous proposez l'achat de biens numériques par carte bancaire, alors qu'il faut passer par le système de paiement Apple. Vous devez alors revoir votre application pour respecter les Guidelines ;
- l'application cite d'autres marques. Par exemple, votre application mentionne Android (Apple n'aime pas du tout cela). Vous devez revoir le contenu de votre application ;
- l'application ou une fonctionnalité ne marchent pas. Par exemple, la connexion échoue. Vous devez corriger le bug.

Si après la lecture du refus motivé d'Apple vous pensez tout de même être dans votre bon droit, n'hésitez pas à répondre à l'équipe de modération en argumentant. Nous avons déjà réussi, par exemple, à faire revenir Apple sur sa décision, car l'équipe de revue n'avait pas testé l'ensemble des fonctionnalités et avait dont manqué certains points importants.

Note

Même si cela n'est pas officiel, une rumeur circule, arguant que chaque compte développeur Apple dispose d'un score, qui évolue notamment en fonction des acceptations ou refus des applications, et qui intervient dans la priorisation des revues. Un compte qui publie des applications régulièrement refusées verra ses temps de revue allongés. De même, un compte avec un bon score pourra a priori plus facilement demander une revue expéditive le moment venu.

Déploiement ad hoc

Une application iOS peut prendre trois formes :

- une application de développement qu'on exécute dans un simulateur ou sur un appareil connecté à Xcode ;
- une application de production qui est envoyée à Apple et qui sera téléchargée par les utilisateurs ;
- une application ad hoc qui peut être envoyée à un sous-ensemble d'utilisateurs sans passer par l'App Store.

L'utilisation d'une application ad hoc est le seul moyen de faire tester votre application à une personne dont vous ne pouvez pas connecter son appareil à votre ordinateur (par exemple un client). Votre application pourra ainsi être « envoyée » aux testeurs que vous aurez choisis. Apple limite le nombre de testeurs à 100 différents au maximum par an, évitant ainsi de faire de ce système un outil de distribution parallèle de votre application. En plus de cela, le profil ad hoc a une durée de vie relativement courte.

Astuce

À la fin de chaque année d'abonnement, vous disposez de quelques jours pour revoir votre liste d'appareils autorisés, dans la limite de 100 différents par an. Si vous n'avez pas atteint la limite des 100, vous pourrez bien sûr en ajouter de nouveaux tout au long de l'année.

Copyright © 2015 Eyrolles.

Pour distribuer une application en mode ad hoc, vous devez ajouter autant d'appareils que de testeurs dans l'interface du Member Center. Vous pouvez également utiliser un outil comme TestFlight, qui est une suite complète d'outils de test pour Apple. Ce service proposait il y a encore quelques mois des outils de tests sur Android, avant qu'Apple ne rachète la société et cesse les activités de test pour Android. Son interface est relativement intuitive et il existe de nombreuses documentations à son sujet, nous n'allons donc pas détailler ici la procédure complète. Sachez simplement qu'en créant un projet sur le service, et en invitant vos testeurs au sein de votre équipe, vous serez en mesure de récupérer leur identifiant unique.

Une fois que vous avez récupéré ces identifiants et créé un appareil pour chacun d'eux, vous pouvez mettre à jour ou concevoir un *provisioning profile* de type ad hoc dans l'interface du Member Center. À l'étape de configuration, pensez bien à sélectionner tous les appareils de vos testeurs.

Vous pourrez ensuite vous rendre dans Xcode, et suivre les mêmes étapes que celles vues précédemment concernant la soumission de votre application à Apple, en sélectionnant toutefois l'option ad hoc au moment opportun. Ce choix vous permettra de télécharger sur votre ordinateur un fichier IPA comprenant votre application.

Astuce

Si Xcode ne semble pas trouver votre profil ad hoc, référez-vous à l'astuce précédente concernant le profil de production et suivez les mêmes étapes.

Ce fichier IPA représente votre application et peut être envoyé à vos testeurs, soit par le biais de TestFlight, soit par le biais d'autres services comme Diawi par exemple (http://www.diawi.com/). Vos testeurs pourront alors tester votre application, sans passer par l'App Store, et sans se brancher à votre ordinateur !

Publication sur le Google Play Store

Création de l'APK

Alors qu'une application iOS prend la forme d'un fichier IPA, une application Android prend la forme d'un fichier APK. Celui-ci peut être généré directement à partir d'ADT, et envoyé à Google via la console Développeur comme décrit plus loin.

Lorsque votre application est prête et a été testée sur des appareils virtuels (via GenyMotion), ainsi que des appareils physiques, vous pouvez passer à l'étape de création de votre APK. Pour cela, effectuez un clic droit sur votre projet dans Eclipse ADT, et cliquez sur Export. Choisissez l'export de type Android>Export Android Application.

Copyright © 2015 Eyrolles.

Figure 7-14
Export d'un APK dans ADT

Vérifiez que le projet sélectionné est le bon et passez à l'étape suivante, si la mention « No errors found. Click Next » apparaît.

Figure 7-15
Choix du projet à exporter

Sélectionnez maintenant l'option Create new keystore. Choisissez un emplacement et un mot de passe.

Copyright © 2015 Eyrolles.

Note

Conservez dans un endroit sécurisé votre clé, ainsi que le mot de passe associé. Si vous les perdez, vous ne serez plus en mesure de mettre à jour votre application.

Figure 7-16
Création de la clé

Remplissez à présent les champs nécessaires à la création de votre clé et validez pour passer à l'étape suivante.

Figure 7-17
Choix de la clé

Copyright © 2015 Eyrolles.

La dernière étape consiste à choisir l'emplacement de création de l'APK.

Figure 7-18
Choix de l'emplacement de l'export

Voilà, votre premier APK de production est prêt !

Création et mise en ligne de l'application

Voyons maintenant comment publier votre première application sur le magasin d'applications d'Android.

Pour cela, connectez-vous à la Developer Console (https://play.google.com/apps/publish/). C'est sur cet espace en ligne que vont se dérouler toutes les étapes.

Pour créer une nouvelle application, cliquez sur le bouton Ajouter une nouvelle application, à droite du titre.

La première étape consiste à choisir la langue de l'application, ainsi que son titre. Vous avez ensuite le choix entre envoyer tout de suite l'APK, c'est-à-dire le code de l'application, ou commencer par créer sa fiche. Choisissez la seconde option.

Figure 7-19
Nouvelle application Google Play

Le processus d'ajout d'une application se compose de cinq étapes : les informations générales, les éléments graphiques, la classification, les coordonnées du développeur et les règles de confidentialités.

Copyright © 2015 Eyrolles.

Informations sur le produit

Cette première étape consiste à fournir les informations générales de l'application : le titre, la description courte et la description complète. Choisissez avec attention ces éléments, car ils seront le premier lien entre votre application et vos futurs utilisateurs.

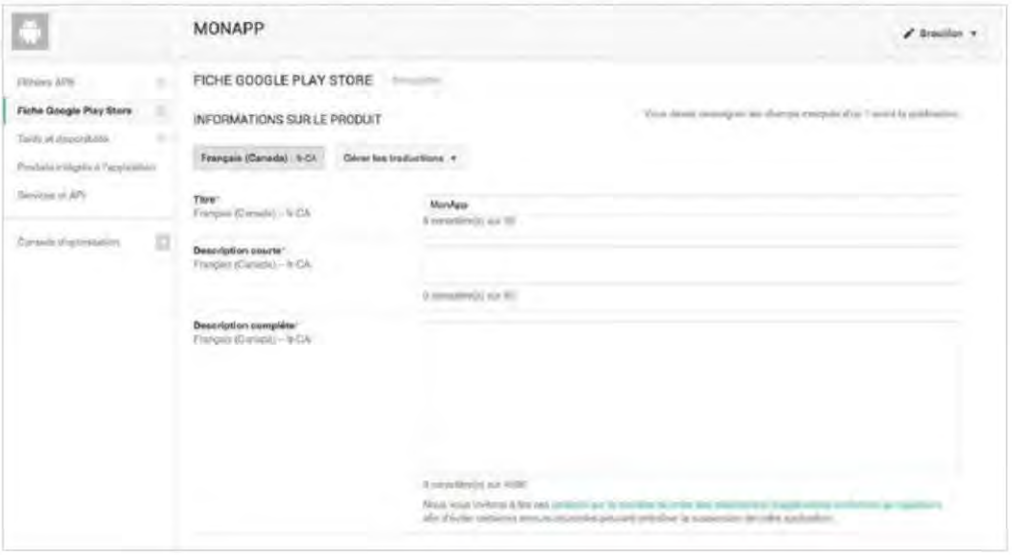


Figure 7-20
Informations sur le produit

Éléments graphiques

Vous devez à présent fournir les icônes et écrans de chargement de présentation de votre application. L'interface est très bien construite et vous présente clairement les différents formats attendus.

Figure 7-21
Éléments graphiques

Copyright © 2015 Eyrolles.

Classification

Puis vous devez choisir le type d'application (une application ou un jeu) que vous souhaitez ajouter, sa catégorie et la classification de son contenu en termes d'audience.

Figure 7-22
Classification

Coordonnées

Fournissez simplement les coordonnées du développeur, avec un site Internet, un e-mail et un numéro de téléphone.

Règles de confidentialité

Vous pouvez dans cette section préciser une URL pointant vers les règles de confidentialité de votre application. Si vous n'en disposez pas, cochez la case adéquate.

L'étape suivante consiste à choisir le tarif et la disponibilité de votre application.

Tarifs

Tout comme sur l'App Store d'Apple, vous pouvez ici choisir de distribuer gratuitement votre application. Si vous décidez de la rendre payante, vous devrez rattacher un compte marchand à votre console développeur.

Copyright © 2015 Eyrolles.

Figure 7-23
Tarifs

Disponibilité

Vous pouvez choisir dans quels pays sera disponible votre application. Cochez « Sélectionner tous les pays » pour que votre application soit disponible partout.

> **Astuce**
> Gardez à l'esprit qu'un public francophone peut être présent dans tous les pays du globe !

Lorsque vous avez renseigné les différents écrans, vous pouvez revenir à l'étape 1 d'envoi du fichier APK. Dans l'onglet Production, cliquez sur Importer votre premier fichier APK en version production. Une fenêtre s'ouvre alors qui vous permettra de glisser-déposer le fichier APK ou d'en sélectionner un sur votre disque.

Figure 7-24
Import d'un APK

Copyright © 2015 Eyrolles.

Figure 7-25
Fenêtre d'import d'un APK

Une fois que votre application est prête, le menu en haut à droite devrait passer de Brouillon à Prêt à être publié. Cliquez sur ce bouton et sélectionnez Publier cette application.

Astuce

Sur l'écran d'envoi de votre APK de production, vous remarquerez deux autres onglets Tests Bêta et Tests Alpha. Ils vous permettent de mettre en ligne des APK de tests pour votre communauté de testeurs, pouvant prendre la forme d'un groupe Google ou d'une communauté Google+. À ces outils, s'ajoute la méthode la plus simple qui consiste à envoyer par e-mail votre APK à vos testeurs.

Maintenant que vous maîtrisez la publication d'applications pour iOS et Android, penchons-nous dans le dernier chapitre sur l'outil PhoneGap Build, qui permet de compiler vos applications dans le Nuage.

Copyright © 2015 Eyrolles.

Copyright © 2015 Eyrolles.

8

PhoneGap Build

Présentation

PhoneGap Build est la plate-forme de compilation d'applications dans le Nuage, signée Adobe. Elle permet à quiconque ne possède pas le matériel adéquat, ou ne souhaite pas installer chacun des SDK requis, de produire les fichiers nécessaires à la publication sur les magasins d'applications en ligne. Le service est gratuit pour la gestion d'une application privée et d'une infinité d'applications open source, généralement non destinées à la vente. Un accès payant est également proposé, celui-ci offrant la gestion simultanée de 25 applications privées pour moins de dix dollars américains par mois. Bien que moins courante et principalement destinée aux entreprises, une offre personnalisée peut être envisageable si cette limite est jugée trop basse.

Figure 8-1
Les différents types de compte PhoneGap Build

Copyright © 2015 Eyrolles.

Chaque abonnement inclut aussi l'ajout de collaborateurs sur des projets d'application : c'est-à-dire l'invitation de toute personne possédant un compte PhoneGap Build à intervenir dans la gestion du cycle de compilation. Évidemment, les droits d'accès sont paramétrables, ainsi certains intervenants pourront déclencher de nouveaux *builds*, tandis que d'autres auront un rôle plus ou moins réduit à celui de bêta-testeur.

À ce propos, PhoneGap Build offre une fonctionnalité inédite nommée Hydration permettant de déployer des mises à jour d'une application donnée sans avoir à rejouer son processus d'installation. Chaque testeur est alors simplement notifié et invité à ne télécharger que les fichiers nécessaires depuis l'application elle-même. Pratique et efficace !

À l'instar de Cordova, le comportement de PhoneGap Build est configuré par le biais d'un fichier `config.xml`. Celui-ci est généralement situé à la racine du répertoire contenant le code source de l'application. À côté du fichier `index.html`, point d'entrée du programme. À ce stade, peu importe la structure accueillant chacun des autres fichiers nécessaires. Notez simplement que tout code natif (.h, .m, .java, etc.) accompagnant ceux-ci sera au mieux ignoré, au pire provoquera l'échec du *build*.

Il va de soi qu'utiliser un tel service de compilation dans le Nuage, aussi pratique soit-il, ne peut offrir autant de possibilités qu'un SDK présent sur une machine dont vous auriez le contrôle total. Parfois vous devrez peut-être effectuer des opérations spécifiques sortant du cadre de PhoneGap Build, il faudra alors s'en passer et compiler soi-même via les utilitaires en ligne de commande Cordova/PhoneGap. La liste des plug-ins pris en charge par le service est, par exemple, relativement restreinte, même si constamment étoffée par les développeurs Adobe et la communauté.

Effectivement, à moins de posséder un compte prémium, il est impossible d'adjoindre n'importe quel plug-in à une application. Seuls les accès à un catalogue vérifié et au registre `plugins.cordova.io` sont permis. C'est d'ailleurs là l'unique façon de compiler du code natif additionnel via PhoneGap Build. Cette précision est importante, car de fait, certaines applications ne pourront tout simplement pas en tirer parti. Le choix des fonctionnalités proposées est donc ici déterminant, aussi bien au départ que dans une optique d'évolution.

Autre point important, le poids total des applications compilées via PhoneGap Build est limité à 50 Mo pour les comptes gratuits, contre 100 Mo pour ceux qui sont payants, et 1 Go pour les membres Adobe Creative Cloud (http://www.adobe.com/fr/creativecloud.html). De plus, ni l'utilisation des versions de Cordova/PhoneGap antérieures à 3.0, ni la compilation pour les plates-formes autres qu'iOS, Android et Windows Phone 8 ne sont permises. Choisir PhoneGap Build, c'est aussi accepter de dépendre à la fois d'une connexion à Internet et de l'état du service, celui-ci pouvant parfois s'avérer inaccessible, comme lors d'éventuelles maintenances et/ou problèmes techniques.

La création d'un compte PhoneGap Build va de pair avec celle d'un identifiant Adobe, l'idée étant de promouvoir les produits estampillés Creative Cloud. Vous le verrez dans la suite de ce chapitre, il est possible de lier un compte GitHub avec PhoneGap Build. Nous encourageons d'ailleurs cette pratique afin que le dialogue entre ces deux services soit optimal. En effet, chaque nouvelle application sera ajoutée depuis un dépôt Git hébergé sur GitHub. Celles qui sont privées pourront également être fournies manuellement en tant qu'archives au format `.zip` – cette solution n'est cependant pas conseillée, car trop fastidieuse. Utiliser GitHub, comme abordé précédemment, présente certains avantages indéniables pour le travail en équipe. Notamment la gestion des *wikis*, *issues* (interfaces graphiques présentant les bugs et autres demandes d'améliorations remontées par la communauté) et autres outils extrêmement pratiques et bien conçus.

Copyright © 2015 Eyrolles.

Figure 8-2
Création d'un identifiant Adobe

Comme toute application Cordova, la configuration du comportement attendu après compilation s'effectue par le biais d'un fichier nommé `config.xml`. Pour des raisons évidentes détaillées ci-après, PhoneGap Build apporte toutefois son lot de différences et autres réglages spécifiques.

Configuration via le fichier config.xml

Paramètres généraux

Avant de prétendre compiler une application, il est nécessaire de connaître le fonctionnement du fichier `config.xml` sur le bout des doigts. Ce fichier indispensable apporte toutes les précisions dont PhoneGap Build a besoin pour arriver au résultat attendu. Voici son contenu minimal :

```
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns="http://www/w3.org/ns/widgets"
  xmlns:gap="http://phonegap.com/ns/1.0"
  xmlns:android="http://schemas.android.com/apk/res/android"
  id="com.reverse.domain"
  version="1.0.0">
  <name>Mon application</name>
  <description>Une description</description>
</widget>
```

Les trois premières lignes définissent un fichier XML widget standard (http://www.w3.org/TR/widgets/) dans lequel les espaces de nommage `gap` et `android` sont autorisés. L'attribut `id` représente quant à lui l'identifiant unique de l'application sur le système d'exploitation mobile. Primordial, cet identifiant est généralement exprimé au format « nom de domaine inversé » – ce qui le rend unique – comme à la cinquième ligne du code qui précède. Suivant de préférence la spécification SemVer (*Semantic Versioning,* http://semver.org/), le numéro de version de l'application permet à l'utilisateur de connaître celle installée sur son système, mais aussi de savoir si une mise à jour est disponible. Enfin, le nom et la description faciliteront tous deux la recherche de l'application sur le portail PhoneGap Build, dans le cas où vous en posséderiez un grand nombre.

Copyright © 2015 Eyrolles.

Astuce

Le nom sera aussi celui affiché sous l'icône sur l'écran d'accueil d'un appareil mobile.

Bien qu'on ne puisse définir que ces quelques informations, il est d'usage d'en fournir davantage et ainsi d'étoffer le fichier `config.xml` avec de nouvelles entrées. On commencera par exemple à préciser pour quelles plates-formes on souhaitera une compilation, et ce grâce à la balise `gap:platform` comme suit :

```
<gap:platform name="ios"/>
<gap:platform name="android"/>
<gap:platform name="winphone"/>
```

Si vous n'ajoutez aucune plate-forme, PhoneGap Build compilera par défaut pour toutes celles prises en charge. Comme être explicite est souvent préférable, nous ajouterons les trois, de même pour les préférences communes suivantes :

```
<preference name="phonegap-version" value="3.6.3"/>
<preference name="orientation" value="default"/>
<preference name="fullscreen" value="false"/>
```

Celles-ci correspondent dans l'ordre à la version de PhoneGap utilisée (voir la liste officielle[1], toutes ne sont pas prises en charge), aux orientations d'écrans autorisées (`landscape`, `portrait` ou `default` pour les deux), ainsi qu'à l'exécution de l'application en plein écran, sans barre d'état (`true` ou `false`).

Astuce

Les jeux vidéo tirent constamment parti de cette fonctionnalité.

Selon les plates-formes, il existe un certain nombre de préférences spécifiques, toutes optionnelles, permettant des réglages plus fins. Elles correspondent dans l'ensemble à celles utilisées par Cordova dans un fichier `config.xml` classique. Toutefois, PhoneGap Build en introduit de nouvelles. Un choix tout à fait logique et compréhensible lorsque avoir la main sur le processus de compilation n'est pas permis.

Pour iOS uniquement

Les préférences spécifiques à iOS sont les suivantes :

```
<preference name="target-device" value="universal"/>
<preference name="prerendered-icon" value="true"/>
<preference name="detect-data-types" value="false"/>
<preference name="exit-on-suspend" value="false"/>
<preference name="deployment-target" value="7.0"/>
```

1. http://docs.build.phonegap.com/en_US/configuring_preferences.md.html#Preferences

Copyright © 2015 Eyrolles.

`target-device` (première ligne du code) permet de choisir sur quel type d'appareil l'application pourra être exécutée : un iPhone, un iPad ou les deux. Les valeurs possibles sont respectivement `handset`, `tablet` et `universal`.

`prerendered-icon` (deuxième ligne du code) autorise, si `true`, le système (iOS 6 et moins) à appliquer un effet « glossy » sur l'icône de l'application.

Si `detect-data-types` (troisième ligne du code) est `true`, le système peut transformer ce qu'il estime être par exemple un numéro de téléphone en lien cliquable `tel://`.

Note

Bien que pratique, il s'agit là d'un fonctionnement typique d'une WebView, laissant ainsi l'utilisateur associer l'application à un simple site web. Ceci n'est pas toujours l'effet recherché. Notez qu'il existe une balise HTML `meta` ayant le même effet[32].

`exit-on-suspend` (quatrième ligne du code) autorise ou non l'application à rester active en arrière-plan. S'il est `false`, chaque retour au premier plan provoquera un redémarrage de celle-ci.

`deployment-target` (cinquième ligne du code) interdit simplement l'installation de l'application sur certaines versions d'iOS.

Pour Android uniquement

PhoneGap Build propose davantage de réglages dédiés à Android. Les voici en détail :

```
<preference name="android-minSdkVersion" value="7"/>
<preference name="android-maxSdkVersion" value=""/>
<preference name="android-targetSdkVersion" value="7"/>
<preference name="android-installLocation" value="internalOnly"/>
<preference name="android-windowSoftInputMode" value="stateVisible"/>
<preference name="splash-screen-duration" value="5000"/>
```

Les trois premières lignes de code ci-dessus permettent de déterminer finement sur quelles versions du système l'application pourra être exécutée. Par défaut la valeur minimale et de `7`, correspondant à Android 2.1/Eclair. La valeur maximale sera la plus récente disponible soit aujourd'hui `21`, Android 5.0/Lollipop. `android-targetSdkVersion` spécifie pour quelle version du SDK l'application est optimisée. Si elle n'est pas renseignée, elle est égale à `android-minSdkVersion`.

Note

Le kit de développement Android possède un numéro de version indépendant de celui du système. Faire la liaison entre les deux n'est pas forcément évident. Sachez alors que Google maintient un tableau détaillé présentant les relations entre versions de plate-forme et niveaux d'API.

2. https://developer.apple.com/library/safari/documentation/AppleApplications/Reference/SafariHTMLRef/Articles/MetaTags.html

Copyright © 2015 Eyrolles.

À la différence des actuels terminaux Apple, ceux sous Android disposent bien souvent d'un port pour carte micro SD facilitant l'extension de leur mémoire de stockage. Il est alors logique de vouloir contrôler la destination d'installation de son application (voir la quatrième ligne du code qui précède). Les valeurs acceptées sont `auto` (selon les préférences de l'utilisateur), `preferExternal` (si possible sur une carte mémoire) et `internalOnly`.

`windowSoftInputMode` (cinquième ligne du code qui précède) définit le comportement attendu lorsque le clavier virtuel doit apparaître. Par exemple, `stateVisible|adjustResize` aura pour effet de n'afficher le clavier que si nécessaire tout en adaptant la taille de la fenêtre contenant l'application. Consultez la documentation éditée par Google[3] pour de plus amples informations.

`splash-screen-duration`, seule préférence Android non préfixée, règle la durée en millisecondes pendant laquelle tout éventuel splashscreen sera visible (voir la sixième ligne du code précédent).

Enfin, il existe un attribut nommé `versioncode` à ajouter sur la balise `widget`. Celui-ci est optionnel et doit contenir un entier correspondant à un numéro de version alternatif destiné à être lu par des machines. S'il est utilisé, sa valeur doit être incrémentée pour chaque mise à jour de l'application.

Note

Une bonne pratique est d'utiliser le plug-in Cordova SplashScreen[34] pour masquer manuellement l'écran de chargement au moment opportun. Peu courants sous Android, les écrans de lancement de l'application relèvent cependant d'une question de design ; une considération importante, car les usagers de ce système d'exploitation ne souhaitent pas une simple copie d'expérience utilisateur iOS, et réciproquement.

Personnalisation avancée

Les quelques préférences précédemment listées peuvent suffire dans la plupart des cas. PhoneGap Build offre cependant davantage de libertés. En effet, Cordova est une couche d'abstraction visant à simplifier la production d'applications multi-plates-formes. De ce fait, l'utilisation des réglages simplifiés a pour effet d'écrire la configuration associée dans des fichiers dédiés. À savoir `Info.plist` sous iOS[5] et `AndroidManifest.xml` sous Android[6].

Astuce

`Info.plist` et `AndroidManifest.xml` sont respectivement situés sous `./platforms/ios/nom-de-l-application/` et `./platforms/android/` dans l'arborescence d'un projet Cordova classique.

En compilant à l'aide des outils en ligne de commande Cordova/PhoneGap, les développeurs peuvent directement modifier ces fichiers ; avec PhoneGap Build cela n'est pas le cas. Pour y

3. http://developer.android.com/guide/topics/manifest/activity-element.html#wsoft
4. https://build.phonegap.com/plugins/1274
5. https://developer.apple.com/library/iOS/documentation/General/Reference/InfoPlistKeyReference/Articles/AboutInformationPropertyListFiles.html
6. http://developer.android.com/guide/topics/manifest/manifest-intro.html

Copyright © 2015 Eyrolles.

pallier, le service rend possible la surcharge de toute propriété spécifique grâce à une balise nommée `gap:config-file`. Son fonctionnement est plus ou moins le même que celui de sa grande sœur `config-file`[7], utile dans la création de plug-ins.

D'après l'exemple extrait de la documentation officielle, ajouter `<preference name="orientation" value="portrait"/>` a pour effet d'écrire dans le fichier `Info.plist` :

```
<key>UISupportedInterfaceOrientations</key>
<array>
  <string>UIInterfaceOrientationPortrait</string>
  <string>UIInterfaceOrientationPortraitUpsideDown</string>
</array>
```

Si vous ne souhaitez pas autoriser l'orientation portrait à 360 degrés, utilisez la balise `gap:config-file` de la façon suivante :

```
<gap:config-file platform="ios" mode="replace"
  parent="UISupportedInterfaceOrientations">
  <array>
    <string>UIInterfaceOrientationPortrait</string>
  </array>
</gap:config-file>
```

Ou encore :

```
<gap:config-file platform="ios" mode="delete"
  parent="UISupportedInterfaceOrientations">
  <array>
    <string>UIInterfaceOrientationPortraitUpsideDown</string>
  </array>
</gap:config-file>
```

Si vous ne souhaitez pas employer la balise `preference`, écrivez simplement :

```
<gap:config-file platform="ios" mode="add"
  parent="UISupportedInterfaceOrientations">
  <array>
    <string>UIInterfaceOrientationPortrait</string>
  </array>
</gap:config-file>
```

Outre la précision de la plate-forme ciblée (ici iOS), la balise `gap:config-file` accepte deux autres attributs. L'attribut `parent` indique quelle fonctionnalité modifier, tandis que l'attribut `mode` définit la façon d'agir sur celle-ci. Pour ce dernier, plusieurs choix s'offrent à vous : `add` pour ajouter des données, `replace` pour remplacer l'ensemble des valeurs existantes, `merge` pour fusionner des attributs sur des propriétés déjà présentes et `delete` pour supprimer des éléments.

7. http://docs.phonegap.com/en/3.3.0/plugin_ref_spec.md.html#Plugin%20Specification_config_file_element

Copyright © 2015 Eyrolles.

Utiliser cette fonctionnalité est légèrement plus simple quand il s'agit d'apporter des modifications au fichier `Info.plist`. En effet, son format clés/valeurs est extrêmement simple à appréhender. A contrario, le Manifeste Android est un fichier XML possédant davantage de balises imbriquées, et de propriétés spécifiques. Par conséquent, la valeur de l'attribut `parent` pour Android doit être un sélecteur valide `XPath` (http://fr.wikipedia.org/wiki/XPath). C'est-à-dire, un peu comme en CSS, un langage dont le but est de localiser des éléments précis au sein d'un document. Ainsi, toujours sans utiliser la balise `preference`, nous pourrions écrire :

```
<gap:config-file platform="android" mode="merge"
  parent="/manifest/application/">
  <activity android:screenOrientation="portrait"/>
</gap:config-file>
```

Note

Pour débugger cette fonctionnalité, il suffit d'extraire le fichier `.ipa` compilé (iOS) après avoir changé son extension en `.zip` donnant ainsi accès au fichier `Info.plist` (ce n'est en réalité qu'une archive compressée). Avec Android, il faudra là aussi extraire le contenu du fichier `.apk`, mais on ne pourra le faire qu'à l'aide d'android-apktool (https://code.google.com/p/android-apktool/), un outil en ligne de commande destiné à cet usage.

Icônes et splashscreens

Dans un but de personnalisation, on choisira quelles icônes et splashscreens afficher. Bien qu'optionnelle, cette étape est pourtant vivement conseillée. En effet, si aucune icône n'est référencée dans le fichier `config.xml`, PhoneGap Build ajoutera à la compilation un fichier contenant le logo PhoneGap. Celui-ci sera alors affiché à divers endroits clés des systèmes d'exploitation de chaque plate-forme choisie. A minima, il est donc intéressant d'ajouter une icône, et ce via la balise `icon` : `<icon src="icon.png"/>` où `icon.png` représente un chemin relatif vers le fichier image à utiliser, généralement au format PNG.

Figure 8-3
L'icône et le splashscreen d'une application

Copyright © 2015 Eyrolles.

Évidemment, de par la diversité des plates-formes, des tailles et densités d'écrans des appareils mobiles, une seule icône ne saurait suffire. On emploiera alors plusieurs fois la balise `icon` conjointement à ses attributs `gap:platform`, `gap:qualifier` (anciennement nommé `gap:density`), `width` et `height`. Même chose concernant les splashscreens dont la balise associée se nomme `gap:splash`. Un parfait exemple de l'ajout d'icônes et splashscreens est disponible sur la documentation officielle PhoneGap Build[8].

Note

Dans une application Cordova classique, l'attribut `gap:platform` est remplacé par la balise `platform`. De même pour `gap:qualifier` qui change de nom pour `density`.

Sous iOS, ces caractéristiques sont plus facilement maîtrisables en raison d'un nombre moins important d'appareils disponibles à ce jour. Sous Android, la fragmentation est plus conséquente, c'est pourquoi l'attribut `gap:qualifier` a remplacé `gap:density`, moins souple à l'usage. Vous pourrez en apprendre davantage au sujet des qualificateurs sur la documentation officielle Android[9]. Toujours sous Android, il est vivement conseillé d'utiliser des images au format `.9.png`[10]. Il s'agit de PNG comprenant une bordure d'un pixel aidant le système à savoir comment étirer l'image sans la déformer.

Un outil destiné à produire ces fichiers est livré avec le kit de développement. Celui-ci, nommé simplement draw9patch est situé dans le répertoire `tools` du SDK. Son usage n'est pas forcément des plus ergonomiques, mais il remplit sa fonction sans être superflu. Bien sûr, ce n'est pas le seul disponible, on notera par exemple l'existence de Better 9 Patch[11], voire Nine Patches[12], un éditeur en ligne.

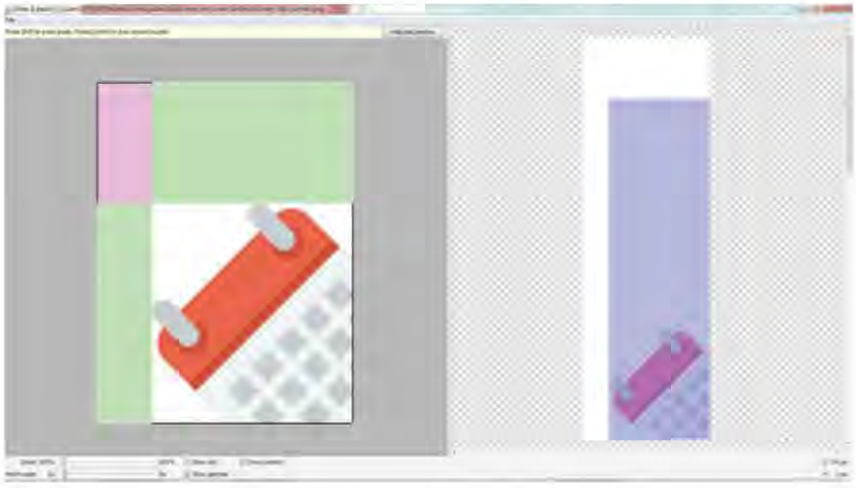


Figure 8-4
L'outil draw9patch du SDK Android

8. http://docs.build.phonegap.com/en_US/configuring_icons_and_splash.md.html#Icons%20and%20Splash%20Screens
9. http://developer.android.com/guide/topics/resources/providing-resources.html
10. http://petrnohejl.github.io/Android-Cheatsheet-For-Graphic-Designers/#nine-patch
11. http://www.roundrect.kr/desktop/better-9-patch/
12. http://romannurik.github.io/AndroidAssetStudio/nine-patches.html

Copyright © 2015 Eyrolles.

Par défaut, comme pour l'icône, PhoneGap Build ira chercher un fichier nommé `splash.png` situé à la racine de l'application (`<gap:splash src="splash.png"/>`). Conserver ce comportement est probablement à éviter puisqu'il sera copié pour chaque plate-forme. Inutile donc de surcharger le paquet téléchargé par l'utilisateur via les magasins d'applications en ligne. N'oublions pas que, même si la 4G est aujourd'hui assez répandue en France, il n'en est pas forcément de même à travers le monde. Par ailleurs, la qualité des réseaux de données mobiles peut varier énormément d'un opérateur et d'un pays à l'autre.

Prenons un exemple concret, Windows Phone 8 ne prend en charge que les splashscreens au format JPG, il serait alors dommage d'inclure une image inutilisée au paquet compilé pour cette plate-forme. De plus, souvenez-vous que PhoneGap Build limite le poids des applications à 50 Mo pour un compte gratuit. Sachez alors qu'il est possible d'inclure un fichier nommé `.pgbomit` (dont le contenu importe peu) dans chaque répertoire que vous souhaitez exclure de l'application compilée. PhoneGap Build aura ainsi l'ordre de ne pas traiter ces dossiers, vous pourrez même y stocker tout code natif si besoin est.

Schémas d'URL

Tout comme pour les liens `tel://`, `mailto://`, `sms://` et `map://`, il est possible de définir des schémas d'URL personnalisés. Ceci afin d'opérer une action particulière lorsqu'un utilisateur accèderait à une adresse correspondante depuis n'importe où dans le système. Malgré un « sandboxing » prépondérant sous iOS, deux applications peuvent alors communiquer entre elles. Par exemple, lorsque vous cliquez sur un lien `mailto://` dans Safari Mobile, l'application Mail est ouverte pour vous laisser rédiger un courrier électronique. De même pour un lien `tel://` qui propose de composer directement le numéro de téléphone associé.

Si vous souhaitez définir votre propre schéma, afin que d'autres applications puissent en tirer parti et vous transmettre différentes informations et données, utilisez la balise `gap:url-scheme` comme ceci :

```
<gap:url-scheme name="com.acme.myscheme" role="None">
  <scheme>myapp</scheme>
</gap:url-scheme>
```

où les attributs `name` et `role` sont tous deux optionnels. La valeur de l'attribut `name` correspond à celle de l'identifiant unique de l'application. Il n'est généralement pas utile de la modifier. Dans le cas contraire, veillez à ce que celle-ci reste unique, sinon la compilation échouera. Le format nom de domaine inversé est, ici encore, tout indiqué. L'attribut `role` quant à lui possède la valeur `None` par défaut. Les autres prises en charge sont `Editor`, `Viewer` et `Shell`. L'utilisation de `gap:url-scheme` se fait conjointement à celle de la balise `scheme` abritant le nom du schéma. Il est possible d'en ajouter plusieurs faisant référence à des actions différentes. Consultez la documentation officielle pour davantage d'informations.

Fonctionnalités

Avant l'éclatement de PhoneGap en un cœur nommé Cordova et une multitude de greffons, il convenait de lister les fonctionnalités utilisées par l'application dans le fichier `config.xml`. En

Copyright © 2015 Eyrolles.

effet, lorsque le code source de PhoneGap contenait encore tout ce qui aujourd'hui est à ajouter en tant que plug-in, il était techniquement difficile de savoir lesquelles allaient réellement servir. Il aurait fallu par exemple « scanner » l'ensemble du code JavaScript à la recherche d'appels aux méthodes exposées par PhoneGap, mais cette façon de faire reste globalement très fragile. Cela principalement car le code peut être minifié pour tenir dans la limite de poids imposée par PhoneGap Build, voire obfusqué. La balise `feature` utilisée à l'époque est aujourd'hui presque totalement obsolète, bien qu'encore prise en charge pour assurer une certaine rétrocompatibilité. La liste des fonctionnalités acceptées est la suivante :

```
<feature name="http://api.phonegap.com/1.0/network"/>
<feature name="http://api.phonegap.com/1.0/camera"/>
<feature name="http://api.phonegap.com/1.0/notification"/>
<feature name="http://api.phonegap.com/1.0/geolocation"/>
<feature name="http://api.phonegap.com/1.0/media"/>
<feature name="http://api.phonegap.com/1.0/contacts"/>
<feature name="http://api.phonegap.com/1.0/file"/>
<feature name="http://api.phonegap.com/1.0/battery"/>
<feature name="http://api.phonegap.com/1.0/device"/>
```

Notez qu'ajouter ou retirer des éléments à cette liste, dans votre fichier `config.xml`, agira seulement sur les permissions au niveau de la configuration des plates-formes. Cela n'est en rien comparable à la gestion des plug-ins qui, elle, permet d'ajouter ou de retirer des fonctionnalités et de gérer les permissions associées dans le même temps.

Seule une fonctionnalité propre à PhoneGap Build et n'ayant aucun rapport direct avec la gestion de permissions empêche réellement le retrait de la balise `feature`. Elle se nomme `debug-server` et autorise l'utilisation d'une instance personnalisée de weinre, c'est-à-dire tournant sur un serveur personnel.

```
<feature name="debug-server" required="true">
   <param name="domain" value="http://debug.custom.com"/>
   <param name="key" value="some_unique_key"/>
</feature>
```

Note

Utiliser cette fonctionnalité permet, par exemple, de réduire les temps de latence réseau, les serveurs d'Adobe étant probablement situés à l'autre bout de la planète.

Plug-ins

Extrêmement important, l'ajout de plug-ins est très simple et peut-être même mieux approché sous PhoneGap Build que par le biais des outils en ligne de commande Cordova/PhoneGap. En effet, on pourra ici, comme NPM via son fichier `package.json` (https://docs.npmjs.com/files/package.json), facilement lister dans le fichier `config.xml` chacun des plug-ins à installer lors de la compilation. C'est-à-dire contrôler finement les versions requises et éviter d'avoir à commiter les plug-ins avec le reste du code de l'application (l'idée étant d'alléger le dépôt pour des temps de clonage plus

Copyright © 2015 Eyrolles.

courts). Seul réel point négatif : à ce jour, le choix de plug-ins sous PhoneGap Build est réduit à deux catalogues : celui du service et celui de la communauté Cordova. Ceci dit, tous deux offrent d'ores et déjà l'accès à une liste conséquente, ouvrant la voie à bien des possibilités !

Certains plug-ins peuvent être présents dans les deux catalogues à la fois. Ceci car le registre PhoneGap Build, seul pris en charge jusqu'en décembre 2014, recense à l'origine ceux proposés par la communauté, mais également vérifiés par l'équipe de développeurs Adobe. Aujourd'hui, on bénéficie d'un choix plus grand grâce au catalogue Cordova. Mais attention, ces plug-ins n'ont pas été contrôlés, il s'agira alors de veiller un minimum à vos choix (couverture de test, prise en charge des plates-formes souhaitées, etc.). Les utilisateurs possédant un compte payant peuvent également soumettre leurs propres plug-ins pour inclusion dans le registre public ou même dans une collection privée.

L'installation s'effectue grâce à la balise `gap:plugin`. Celle-ci possède un attribut `name`, obligatoire, correspondant à un identifiant unique généralement au format nom de domaine inversé. Un attribut `source`, optionnel, permet de préciser dans quel catalogue aller chercher le plug-in. Sa valeur est `pgb` pour PhoneGap Build (par défaut) ou `plugins.cordova.io`. Enfin, il existe un attribut `version`, lui aussi optionnel, dont l'usage est recommandé. S'il est omis, chaque nouvelle compilation ira piocher la version la plus récente du plug-in en question sur le catalogue associé. Or ce fonctionnement n'est probablement pas souhaité, car chaque mise à jour peut apporter son lot de changements significatifs et donc potentiellement provoquer des erreurs au sein de l'application. Dans l'idéal, on verrouillera alors soit une version donnée, testée et approuvée, soit on autorisera uniquement certaines révisions ultérieures.

Astuce

Il est préférable qu'un build soit toujours reproductible, facilitant ainsi la chasse aux éventuelles erreurs. De plus, les développeurs de plug-in ne suivent pas toujours la spécification SemVer.

Voici quelques exemples concrets :

```
<gap:plugin name="com.phonegap.plugin.statusbar"/>
<gap:plugin name="com.phonegap.plugin.statusbar" version="1.1.0"/>
<gap:plugin name="org.apache.cordova.statusbar" source="plugins.cordova.io"
version="~0"/>
<gap:plugin name="org.apache.cordova.inappbrowser" source="pbg" version="~0.5"/>
<gap:plugin name="org.apache.cordova.inappbrowser"  source="plugins.cordova.io"
version="~0.5.4"/>
```

À la première ligne, on installe le plug-in StatusBar[13] depuis le catalogue officiel PhoneGap Build ; toujours dans sa version la plus récente – à éviter.

À la deuxième ligne de code, on verrouille l'installation du plug-in StatusBar, depuis le catalogue PhoneGap Build, dans sa version 1.1.0 – recommandé, quelle que soit la source choisie.

13. https://build.phonegap.com/plugins/505

Copyright © 2015 Eyrolles.

À la troisième ligne, on autorise l'installation d'un autre plug-in StatusBar[14], depuis le catalogue Cordova, dans n'importe quelle version supérieure ou égale à 0.0.0 et inférieure à 1.0.0 – à éviter.

À la quatrième ligne, on autorise l'installation du plug-in InApp Browser[15], depuis le catalogue PhoneGap Build, dans n'importe quelle version supérieure ou égale à 0.5.0 et inférieure à 0.6.0 – encore à éviter.

Enfin, à la cinquième ligne, on autorise l'installation du même plug-in InApp Browser, cette fois depuis le catalogue Cordova[16], et dans n'importe quelle version supérieure ou égale à 0.5.4 et inférieure à 0.6.0 – toujours à éviter.

Dernier point, certains plug-ins, comme GapReload[17], requièrent des paramètres additionnels. Ceux-ci doivent être passés à l'aide de la balise `param` de la façon suivante :

```
<gap:plugin name="pro.fing.cordova.gapreload" source="plugins.cordova.io"
version="1.1.0">
  <param name="SERVER_HOST" value="192.168.0.15"/>
  <param name="SERVER_PORT" value="8080"/>
</gap:plugin>
```

Attention

Il est nécessaire de lire la documentation associée à chaque plug-in pour en comprendre le fonctionnement, ainsi que la procédure d'installation.

Parfois, l'appel de code JavaScript supplémentaire est requis. Si besoin, on inclura par exemple une référence au fichier associé dans la page `index.html`. Cela étant dépendant de l'implémentation, seule sa documentation pourra vous donner la bonne marche à suivre. N'ajoutez jamais le code d'un plug-in dans le répertoire contenant les fichiers sources de votre application, la compilation échouerait ou des problèmes pourraient apparaître à l'usage. Sinon, veillez à utiliser le fichier `.pgbomit`, prévu à cet effet.

Sécurité

Maîtriser les droits d'accès aux ressources distantes est important en termes de sécurité. PhoneGap Build offre exactement les mêmes possibilités qu'une application compilée via les outils en ligne de commande Cordova/PhoneGap. C'est le rôle de la balise `access` et ses mécanismes de liste blanche.

Astuce

Dans l'idéal, on n'utilisera pas de joker/wildcards tels que présents dans la plupart des exemples consultables en ligne. On prendra soin de ne lister que les sources autorisées.

14. http://plugins.cordova.io/#/package/org.apache.cordova.statusbar
15. https://build.phonegap.com/plugins/1169
16. http://plugins.cordova.io/#/package/org.apache.cordova.inappbrowser
17. http://plugins.cordova.io/#/package/pro.fing.cordova.gapreload

Copyright © 2015 Eyrolles.

Configuration via ConfiGAP

Maintenir le fichier `config.xml` est relativement aisé, mais il peut vite devenir suffisamment verbeux pour que sa lecture ne soit pas facilitée, notamment à cause des nombreuses balises `icon` et `gap:splash` accompagnées de tout contenu ayant trait aux différentes plates-formes. Si tel est aussi votre avis, sachez qu'il existe un utilitaire extrêmement pratique nommé ConfiGAP, à télécharger sur http://configap.com/.

Il s'agit d'une application Adobe AIR (http://get.adobe.com/fr/air/), un environnement d'exécution multi-plates-formes principalement dédié aux ordinateurs de bureau et leurs différents systèmes d'exploitation. Adobe AIR propose depuis mars 2008, aux développeurs web/Flash, de créer des applications pouvant accéder à certaines fonctionnalités natives en dehors d'un navigateur Internet. Sorti peu avant les débuts de PhoneGap, il pourrait presque être considéré comme son aîné. Relativement méconnu, Adobe Air n'est pas fourni de base sur la majorité des machines vendues aujourd'hui. Il faudra donc probablement l'installer avant de pouvoir utiliser ConfiGAP.

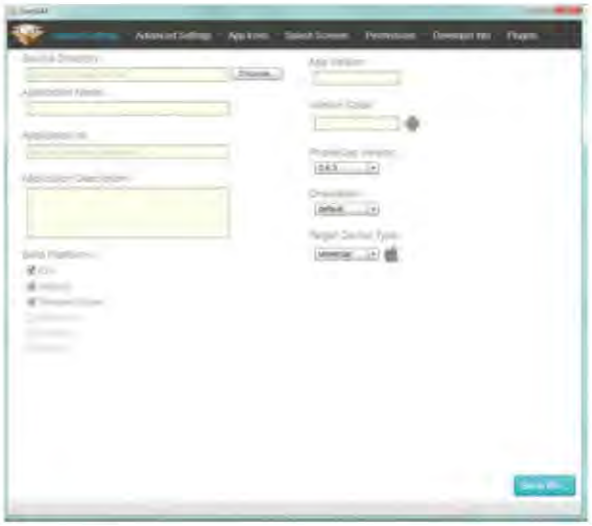
Pour ce faire, cliquez sur le bouton de téléchargement présent sur son site officiel et suivez quelques instructions rapides suffiront à venir à bout du processus. Après quoi, il conviendra d'exécuter le fichier `ConfiGAP.air` pour accéder à l'écran d'accueil de l'application. Si l'interface n'est aujourd'hui disponible qu'en anglais, elle n'en reste pas moins très simple à l'usage. Vous pourrez au départ soit créer un nouveau fichier `config.xml`, soit en éditer un existant. ConfiGAP se découpe en sept onglets présents dans la barre de navigation en haut, chacun accueillant son lot de champs de formulaire pas nécessairement requis.

Paramètres généraux

C'est dans l'onglet General Settings (voir figure 8-5) que vous renseignerez les réglages généraux de votre application tels que son nom, son identifiant unique, sa description et son numéro de version. Vous pourrez choisir également pour quelles plates-formes compiler, quelle version de PhoneGap utiliser, ainsi que les orientations d'écran et le type d'appareil à prendre en charge (pour iOS uniquement). BlackBerry, Symbian et WebOS sont listées pour des raisons historiques, mais ne sont actuellement plus supportées par PhoneGap Build. ConfiGAP ne vous laissera donc pas les sélectionner. Enfin, le champ Version Code, destiné à Android, modifie la valeur de l'attribut `versionCode` de la balise `widget`.

Copyright © 2015 Eyrolles.

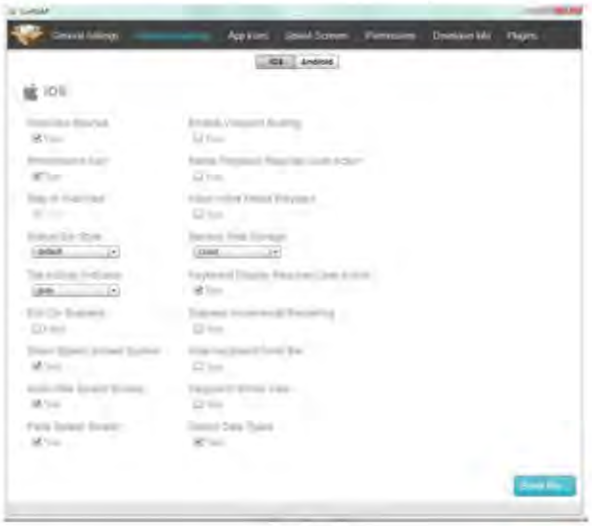
Figure 8-5
L'écran General Settings



Paramètres avancés

L'onglet Advanced Settings regroupe les configurations avancées propres à iOS et Android. Certaines sont présentes uniquement pour une raison de rétrocompatibilité ; d'autres sont aujourd'hui totalement inopérantes. À ce sujet, il est donc préférable de consulter la documentation la plus à jour de PhoneGap Build[18], ainsi que celle de Cordova[19], voire de PhoneGap[20]. Le tableau 8-1 présente les préférences actuellement disponibles dans ConfiGAP ainsi que leurs équivalents dans PhoneGap Build.

Figure 8-6
L'écran Advanced Settings pour iOS



18. http://docs.build.phonegap.com/en_US/configuring_preferences.md.html#Preferences
19. http://cordova.apache.org/docs/en/3.6.0/config_ref_index.md.html#The%20config.xml%20File
20. http://docs.phonegap.com/en/3.6.0/config_ref_index.md.html#The%20config.xml%20File

Copyright © 2015 Eyrolles.

Figure 8-7
L'écran Advanced Settings pour Android

Tableau 8-1. Préférences avancées dans ConfiGAP et équivalents dans PhoneGap Build

iOS (voir figure 8-6)		Android (voir figure 8-7)	
ConfiGAP	**PhoneGap Build**	**ConfiGAP**	**PhoneGap Build**
prerendered Icon	prerendered-icon	Minimum SDK Version	android-minSdkVersion
detect Data Types	detect-data-types	Maximum SDK Version	android-maxSdkVersion
exit On Suspend	exit-on-suspend	Target SDK Version	android-targetSdkVersion
		Install Location	android-installLocation
		SplashScreen Duration	splash-screen-duration

Note

La version de ConfiGAP 1.3.16, présentée dans cet ouvrage, date du 23 novembre 2014 et, comme PhoneGap Build, n'intègre pas encore Cordova 4. Il faudra donc s'assurer de consulter la bonne déclinaison des documentations associées.

Icônes et splashscreens

Pas de surprises, ces onglets vous permettront de choisir les fichiers correspondant aux tailles d'icônes et splashscreens les plus courantes, ceci en fonction des trois plates-formes dominant le marché actuel (iOS, Android et Windows Phone), des versions des systèmes d'exploitation mobiles (typiquement iOS jusqu'à 6 et iOS 7 et plus), puis des différentes densités d'écrans. Souvenez-vous, PNG et .9.png sont ici les formats à privilégier.

Copyright © 2015 Eyrolles.

Figure 8-8
L'écran App Icons pour iOS

Figure 8-9
L'écran de lancement de l'application pour iPhone

Permissions et informations

Dans l'écran Permissions, accompagné des réglages maintenant obsolètes pour Android et Windows Phone, se trouve une interface gérant l'accès aux ressources externes via la balise `access`. La présentation en tableau est particulièrement lisible. L'écran Developer Info, est quant à lui extrêmement simpliste, car il propose uniquement de renseigner un nom, une adresse e-mail et une URL en rapport avec l'auteur de l'application. On pourra choisir de retenir ces informations afin de ne plus avoir à les rentrer par la suite.

Copyright © 2015 Eyrolles.

Figure 8-10
L'écran Permissions

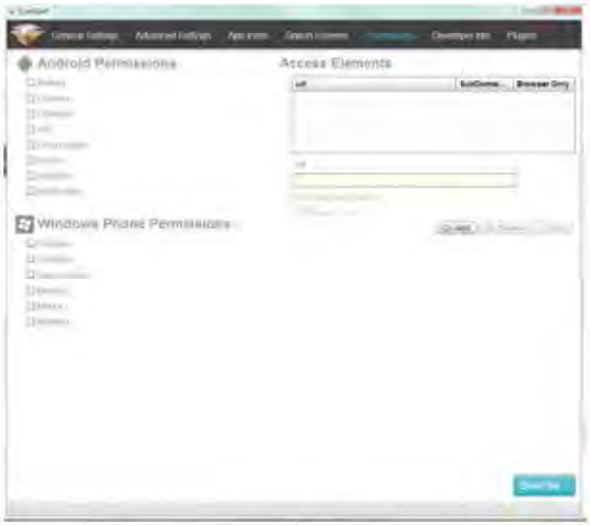
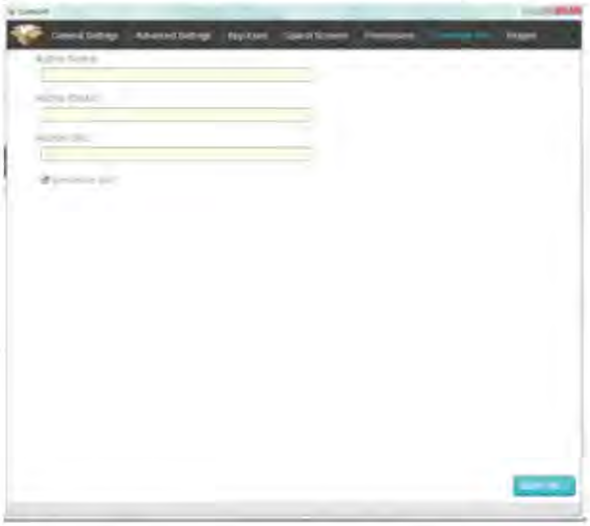


Figure 8-11
L'écran Developer Info

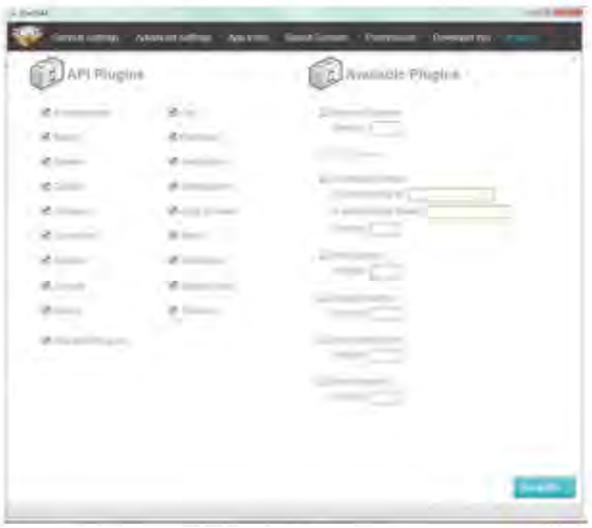


Vue Plugins

Enfin, la vue Plugins offre une gestion rapide et simplifiée des plug-ins couramment utilisés au sein des applications Cordova/PhoneGap. La possibilité d'en ajouter depuis le catalogue Cordova étant récente (début décembre 2014), ConfiGAP n'inclut actuellement pas encore cette fonctionnalité. Le logiciel n'offre pas non plus un accès total au registre vérifié PhoneGap Build. En revanche, les classiques composants, autrefois le cœur de PhoneGap, sont présents : de l'accéléromètre aux vibrations, en passant par l'accès aux contacts et à la géolocalisation.

Copyright © 2015 Eyrolles.

Figure 8-12
L'écran Plugins



On pourra choisir de tous les activer/désactiver simultanément. Sur le papier, cela semble utile, mais dans la pratique il s'avère assez rare qu'une application ait besoin de chacun d'entre eux. On ne cochera alors que ceux vraiment utilisés, car des permissions particulières sont parfois requises et il est préférable d'en demander le moins possible pour ne pas effrayer l'utilisateur.

Note

Un utilisateur devra toujours savoir ce qu'il advient des données personnelles sensibles éventuellement récoltées par l'application, par exemple ses contacts ou sa position géographique.

Pour terminer, ConfiGAP aide également à l'installation et la configuration de plug-ins tiers toujours très utiles, tels que Google Analytics, Push Notifications, ou encore Facebook Connect. La liste est relativement courte, mais il vous sera toujours possible de modifier manuellement le fichier `config.xml` produit pour en ajouter de nouveaux. Un conseil, vérifiez toujours son contenu dans votre éditeur de code préféré afin d'éviter toute erreur. En résumé ConfiGAP est un outil très pratique pour mettre au point une base de configuration rapidement.

Utiliser PhoneGap Build

Maintenant que vous maîtrisez la configuration de PhoneGap Build, il est grand temps de voir comment utiliser le service. Vous apprendrez alors qu'il est possible gérer des applications et leur compilation de différentes manières, principalement via l'interface web dédiée. C'est d'ailleurs par celle-ci que nous débuterons notre chemin. Munissez-vous alors d'une application d'essai – vous pouvez d'ailleurs vous servir du code de celle décrite dans la deuxième partie de cet ouvrage. Pour l'exemple, nous utiliserons une application de démonstration classique hébergée sur GitHub à l'adresse suivante : https://github.com/phonegap/phonegap-start. Pensez aussi à créer votre identifiant Adobe si ce n'est pas encore fait.

Copyright © 2015 Eyrolles.

Via l'interface web

Une fois identifié sur le site web officiel PhoneGap Build, la liste de vos applications est affichée. Si vous n'en avez pas encore, le service vous proposera d'en ajouter une (voir figure 8-13), soit open source, soit privée. Sinon, un bouton + new app vous permettra d'accéder à ce même formulaire.

Figure 8-13
Formulaire d'ajout d'une application

Note

L'interface du service est disponible en français. La personnalisation s'effectue grâce à un menu déroulant présent dans chaque pied de page du site.

Une application open source étant visible par chacun des utilisateurs de la plate-forme, son code doit être hébergé sur GitHub. On aura alors besoin de l'URL du dépôt Git, accompagnée si besoin d'un nom de branche ou de tag, ce qui permettra de travailler sur une version différente de celle située sous *master*. Il en est de même pour une application privée, mais l'on pourra également télécharger une archive au format `.zip` contenant les fichiers nécessaires.

Un conseil, sachez qu'au lieu d'entrer une URL dans le champ dédié, il est possible de lier un compte GitHub à PhoneGap Build afin d'accéder directement à la liste de vos dépôts Git. Cliquez sur le lien « Connect your GitHub account » pour accéder à votre page d'édition de profil utilisateur (voir figure 8-14) et pressez le bouton « Connect a GitHub ID » prévu à cet effet. Cette étape est optionnelle, mais néanmoins recommandée et peut être inversée à tout moment. Ne vous préoccupez pas des autres réglages présents dans l'édition de profil, nous y reviendrons. Notez simplement que vous pouvez modifier votre adresse e-mail et même supprimer votre compte si besoin. Pour revenir à la page précédente, cliquez sur Apps dans la barre de navigation tout en haut.

Copyright © 2015 Eyrolles.

Figure 8-14
La page d'édition de profil utilisateur

Ajouter une application

Ajoutons maintenant une application open source en entrant par exemple : https://github.com/phonegap/phonegap-start.git dans le champ dédié, puis en pressant sur le bouton « Pull from .git repository » situé juste en dessous. Après un certain temps de téléchargement, vous devriez avoir à l'écran l'équivalent de la figure 8-15. Soit un encadré comprenant l'icône de l'application, son nom, sa description et plusieurs boutons importants comme Delete pour arrêter la procédure et Ready to build pour continuer. Dans le cas contraire, un message d'erreur sera affiché, décrivant les raisons ayant conduit à cet échec. Un bandeau d'avertissement peut également être présent si la version de PhoneGap utilisée n'est pas la plus récente, ceci n'est pas gênant. Pressez le bouton Ready to build : vous devriez maintenant avoir sous les yeux l'équivalent de la figure 8-16, soit une vue sommaire de l'état de la compilation.

Figure 8-15
Ajout d'une application

Copyright © 2015 Eyrolles.

Figure 8-16
Vue sommaire

Pour accéder à une vue détaillée, cliquez sur le logo ou le nom de l'application. Plus intéressante, celle-ci correspond à un poste de pilotage complet adaptable selon la taille de votre écran. L'en-tête reprend l'icône, le nom et la description de l'application, ainsi que plusieurs boutons. Parmi eux, Update code sert à fournir de nouveaux fichiers sources de l'application, pour compiler une version différente de celle-ci. Rebuild all, quant à lui, donne l'ordre au service d'initier une compilation pour toutes les plates-formes ciblées.

Figure 8-17
Vue détaillée

En cliquant sur Install, on accédera soit à une page recensant tous les téléchargements de paquets disponibles, soit directement au paquet nécessaire depuis un appareil mobile compatible. Un bouton Share, visible seulement pour les applications publiques, permet de naviguer sur une page de partage accessible à tous. Enfin, point important, la présence d'un code QR (http://fr.wikipedia.org/wiki/Code_QR), offre une procédure d'installation facilitée, nécessitant toutefois un lecteur dédié sur vos appareils mobiles.

Copyright © 2015 Eyrolles.

Astuce

Il en existe un nombre particulièrement conséquent, aussi bien pour iOS qu'Android, une simple recherche avec les mots clés « QR Code » vous renverra les plus populaires.

Le fonctionnement attendu est toujours le même, on se servira de l'appareil photo pour scanner le code en question et être redirigé vers le lien d'installation adéquat. Pratique !

L'onglet Build

Sous l'en-tête se situe une barre d'onglets. Le plus important, d'entre eux, Build, est ouvert par défaut. Il fait état de la compilation pour chacune des plates-formes choisies. Lorsqu'elle est terminée, celle-ci pourra être relancée indépendamment à l'aide du bouton Rebuild correspondant.

Log donne accès aux traces obtenues en sortie du terminal lors de l'exécution des outils de compilation. Le dernier bouton, tout à droite, sert à télécharger le paquet compilé ou bien à afficher plus d'informations concernant d'éventuelles erreurs. Enfin, le menu déroulant à côté du logo de chaque plate-forme, permet de choisir une clé avec laquelle signer l'application pendant la compilation. Pour iOS, sélectionner une clé valide est obligatoire.

Évidemment, si vous utilisez PhoneGap Build pour la toute première fois, vous devez d'abord ajouter les clés nécessaires. Elles sont un élément de sécurité primordial destiné à apposer votre empreinte numérique, unique sur chacune de vos applications. Il est alors important que celles-ci restent en votre possession et soient cryptées et protégées par un mot de passe fort. Rassurez-vous, les télécharger ici reste toutefois relativement sans danger. Adobe est une entreprise sérieuse et applique une politique très stricte quant à l'accès et à l'utilisation de ces dernières ; aucun mot de passe n'est d'ailleurs stocké sur le service. Si vous gérez vos clés avec suffisamment d'attention, tout risque d'usurpation d'identité devrait alors être extrêmement faible.

Note

Adobe n'envoie pas de clé sur demande. Pensez donc à bien conserver des copies au cas où.

Générer des clés pour iOS

PhoneGap Build attend que vous combiniez un certificat et un pro profil de provisionnement, afin d'associer une provenance à une destination. C'est-à-dire des informations sur le développeur et une liste des appareils autorisés à installer l'application. S'il existe typiquement deux types de profils de provisionnement (développement et distribution), vous souhaiterez certainement créer/ajouter tout autant de clés. Ceci étant, les fichiers fournis par Apple sur le Developer Center ne sont pas directement pris en charge. Il faudra les crypter dans un format sécurisé dont l'extension est `.p12`.

Copyright © 2015 Eyrolles.

Pour cela, plusieurs méthodes sont possibles. Si vous possédez un Mac, vous pouvez effectuer cette conversion dans l'application Trousseau d'accès[21]. Sinon, ouvrez une nouvelle fenêtre de terminal dans le répertoire où se situent vos certificats et clés privées. Entrez `which openssl`, ce qui retournera le chemin d'accès à `openssl` si l'exécutable est accessible depuis le `PATH`. Si aucun chemin ne vous est renvoyé, il y a de grandes chances pour que l'outil ne soit pas présent sur votre système. Vous pourrez alors suivre les instructions présentes sur le site officiel (https://www.openssl.org/source/), utiliser un gestionnaire de paquets, ou toute autre méthode de votre choix pour l'installer. De même, si l'exécution des commandes suivantes provoque des erreurs, il est possible que votre version d'openssl mérite une mise à jour.

Vous allez maintenant avoir besoin de fichiers `.cer` et `.key` pour continuer. Si vous n'en possédez pas encore, référez-vous au chapitre précédent, à la section « Gestion des certificats sur Member Center » (page 130), ou exécutez les commandes suivantes (pensez à changer les valeurs passées en paramètres) :

```
openssl genrsa -des3 -out mykey.key 2048
```

pour générer le fichier `.key`, associé, lorsqu'on vous le demandera, à un mot de passe fort et :

```
openssl req -new -key mykey.key -out ios.csr -subj "/emailAddress=MY-EMAIL-ADDRESS,
CN=COMPANY-NAME, C=COUNTRY-CODE"
```

pour créer un fichier `.csr` qu'il faudra ensuite uploader sur le portail développeur iOS dans le but de récupérer le fichier `.cer` requis.

La création d'un fichier `.p12` à partir de fichiers `.cer` et `.key` s'opère alors en deux étapes. Premièrement, entrez :

```
openssl x509 -in developer_identity.cer -inform DER -out developer_identity.pem
-outform PEM
```

en prenant soin de modifier `developper_identity` si besoin. Puis exécutez :

```
openssl pkcs12 -export -inkey mykey.key -in developer_identity.pem -out iphone_dev.p12
```

en adaptant les noms de fichiers à votre situation. Entrez le mot de passe choisi pour le fichier `.key` à chaque fois que nécessaire et choisissez-en un différent pour l'export (utilisation sur PhoneGap Build). Félicitations ! Vous possédez maintenant le fichier `.p12` tant convoité. Gardez celui-ci à portée de main, accompagné de vos différents profils de provisionnement.

Copyright © 2015 Eyrolles.

21. http://docs.build.phonegap.com/en_US/signing_signing-ios.md.html#iOS%20Signing

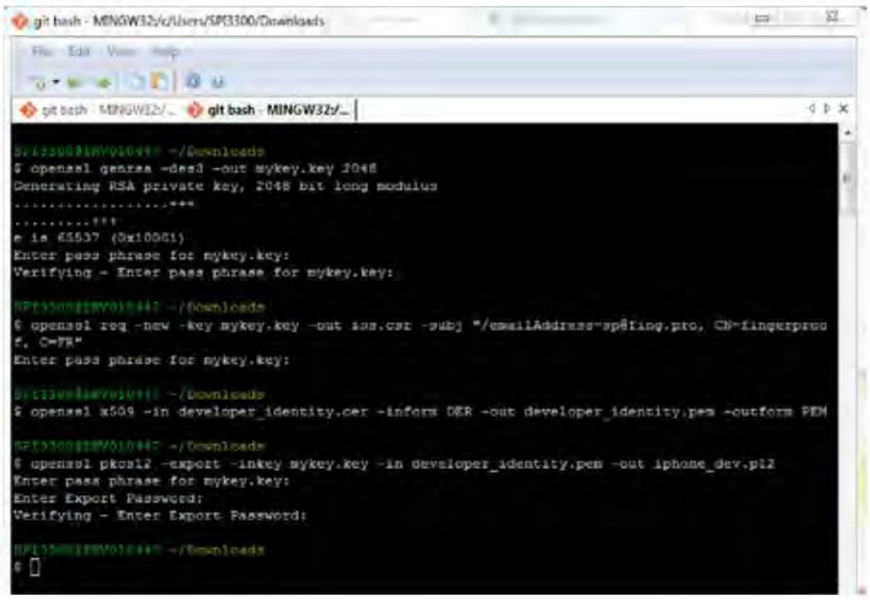


Figure 8-18
Génération d'une clé pour iOS

Générer des clés pour Android

Si Android ne nécessite réellement que la signature des applications distribuées sur le Google Play Store, il est préférable de créer également une clé de développement. On utilisera alors l'outil `keytool`, fourni avec Java. Vérifiez que celui-ci est inclus dans votre `PATH` en entrant `which keytool` dans une fenêtre de terminal. Sinon, référez-vous au chapitre 2 pour installer Java. Dans le répertoire de travail de votre choix, exécutez :

```
keytool -genkey -v -keystore keystore_name.keystore -alias alias_name -keyalg RSA
-keysize 2048 -validity 10000
```

où `keystore_name` et `alias_name` (utile pour faire référence à la clé par la suite) sont à ajuster selon vos préférences. Puis suivez les instructions à l'écran vous invitant à entrer diverses informations vous concernant (voir figure 8-19). L'utilitaire vous proposera ensuite de choisir un mot de passe différent pour l'alias. Cette mesure est optionnelle, bien que davantage sécurisée. Pressez simplement la touche Entrée de votre clavier. Félicitations ! Vous venez de créer la clé nécessaire (`.keystore`) dans le répertoire de travail.

Copyright © 2015 Eyrolles.

Figure 8-19
Génération d'une clé pour Android

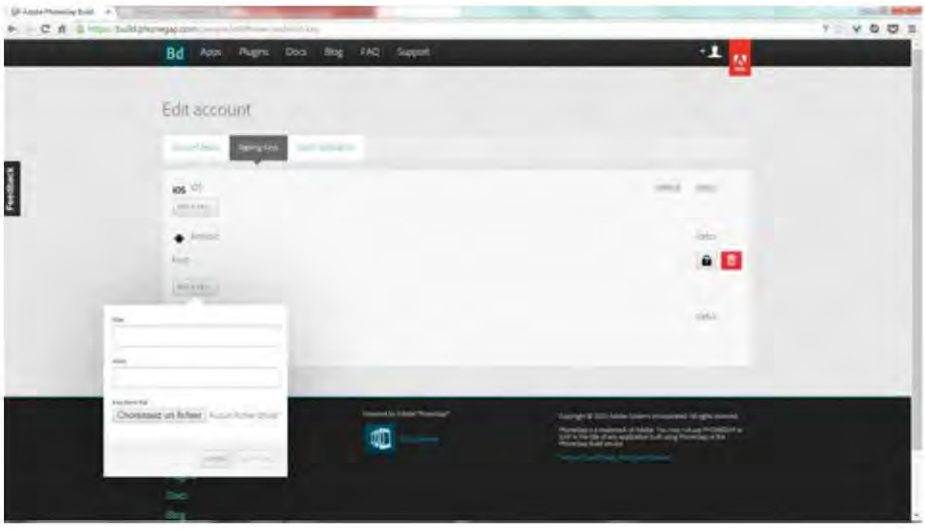
Télécharger des clés

De retour sur l'interface web PhoneGap Build, rendez-vous dans la page d'édition de profil ; sous l'onglet Signing Keys. Téléchargez maintenant chacune des clés générées précédemment (voir figure 8-20). Le champ title, commun à toutes les plates-formes, correspond au nom de votre clé.

Astuce

Choisissez, par exemple, un nom précis tel que `Dev - iOS - app.bundle.id`. Cela vous facilitera la recherche par la suite. Attention, il est pour l'instant impossible de modifier les détails d'une clé a posteriori.

Figure 8-20
Télécharger une clé



Copyright © 2015 Eyrolles.

Vous remarquerez ensuite l'apparition de boutons comprenant l'icône d'un cadenas. En effet, PhoneGap Build verrouille automatiquement l'utilisation de chaque clé après leur ajout et une heure passé leur déverrouillage. Cliquez sur ces boutons, entrez le ou les mots de passe associés aux clés en question. Si vous aviez choisi des mots de passe d'alias différents lors de la génération de vos clés Android, ceux-ci sont à écrire dans le champ Certificate password. Relancez enfin la compilation de votre application. Patientez. Vous pourrez ainsi vérifier le bon fonctionnement des clés. Sinon, suivez les instructions du message d'erreur associé pour résoudre le problème.

Plugins, Collaborators et Settings

Passons maintenant aux autres onglets de l'interface web PhoneGap Build. Plugins recense tous les plug-ins utilisés dans votre application. Cette vue présente également le code que vous auriez dû ajouter dans votre fichier `config.xml`, afin de verrouiller leurs différentes versions. Peu utile si vous avez suivi nos recommandations à ce propos.

Dans l'onglet Collaborators, vous pouvez inviter quiconque à intervenir dans le cycle de développement de votre application. Soit en tant que développeur possédant les droits en lecture et écriture sur l'interface PhoneGap Build, soit en tant que simple testeur ayant un accès réduit. Pour ce faire, entrez simplement l'adresse e-mail d'une de vos connaissances dans le formulaire dédié, et validez. Le service lui enverra alors une invitation. Si cette personne n'est pas membre, elle pourra s'inscrire directement en suivant un lien dans le message en question. Sinon, votre application apparaîtra sur son compte. Vous pourrez évidemment révoquer tout accès à chaque fois que cela s'avérera nécessaire.

Figure 8-21
L'onglet Collaborators

Pour terminer, l'onglet Settings reprend quelques-unes des informations présentes dans le fichier `config.xml`. Il ne vous sera pas permis de les modifier sans en télécharger une version différente. Vous pourrez en revanche choisir de supprimer l'application, de la rendre privée, de modifier l'adresse pointant vers son dépôt Git ou encore d'activer les fonctionnalités nommées Debugging et Hydration.

Copyright © 2015 Eyrolles.

Figure 8-22
L'onglet Settings

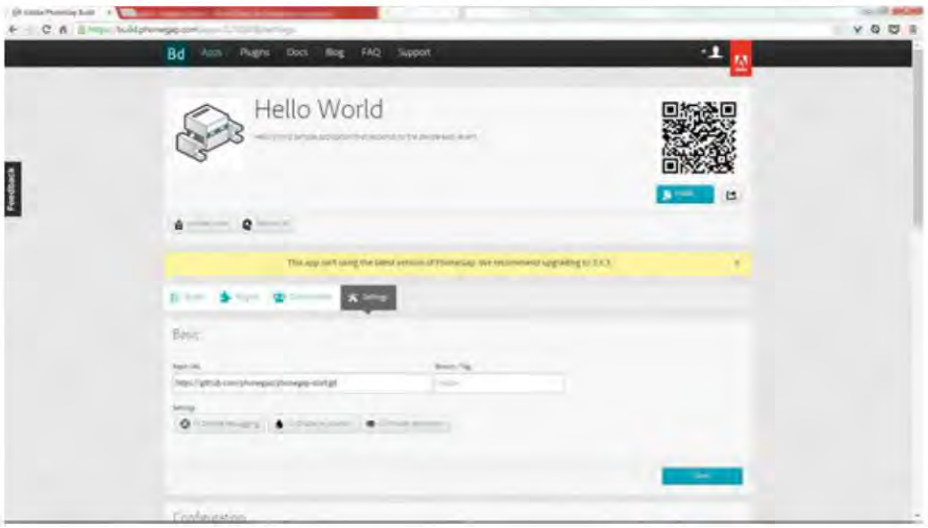


Figure 8-23
Vue détaillée complète

Debugging et Hydration

PhoneGap Build, en plus d'offrir la compilation d'applications dans le Nuage, comprend également deux fonctionnalités très utiles. La première, nommée Debugging[22], si activée, associe une instance de l'outil weinre tournant soit sur les serveurs d'Adobe, soit sur la machine de son choix. Grâce à elle, il sera possible, comme vous l'avez vu précédemment, d'accéder à une interface proche des outils de développement de Google Chrome pour débugger vos applications.

22. http://docs.build.phonegap.com/en_US/debugging_local_debugging_tools.md.html#Local%20Debugging%20Tools

Copyright © 2015 Eyrolles.

Note

Il est préférable de procéder comme pour une application Cordova/PhoneGap classique, à savoir effectuer le plus gros du développement en local dans son navigateur web, puis utiliser la console JavaScript intégrée à weinre si besoin.

Après l'activation/désactivation de Debugging, une compilation et une nouvelle installation de l'application sont nécessaires. L'accès à la fonctionnalité s'effectue alors grâce à un simple bouton Debug, désormais disponible dans l'en-tête du service.

Figure 8-24
weinre sur PhoneGap Build

La seconde, Hydration, a été créée pour faciliter le déploiement des dernières versions d'une application sur autant d'appareils mobiles que nécessaire, mais aussi pour réduire les temps de compilation en général. Une fois activée, cette fonctionnalité nécessitera une ultime installation manuelle pour être opérationnelle. Après quoi, la mise à jour sera simplement poussée directement vers toutes les machines concernées. À chaque démarrage de l'application en question, une boîte de dialogue apparaîtra alors si une nouvelle version est disponible. L'utilisateur est invité à ne télécharger que les fichiers modifiés (ou bien à continuer, jusqu'au prochain redémarrage ou « tap » à trois doigts) à utiliser ceux présents dans la mémoire de l'appareil. Une connexion à Internet est obligatoire dans ce cas.

Hydration pourra être désactivée à tout moment, mais nécessitera alors une nouvelle installation manuelle. Attention, elle n'est pas destinée à court-circuiter les processus de soumission et revue des magasins d'applications. Elle est seulement utile à des fins de test dans le cadre du cycle de développement. Essayer de télécharger, par exemple, une telle application sur l'App Store se soldera très probablement par un rejet de la part d'Apple.

Copyright © 2015 Eyrolles.

Figure 8-25
Mise à jour d'une application utilisant Hydration

Via l'API REST

PhoneGap Build, en plus d'être extrêmement utile, offre une API REST[23] *(Representational State Transfer)* dont l'usage est destiné à quiconque souhaiterait mettre au point sa propre interface, automatiser certaines actions, voire inclure totalement le service à son environnement de développement. Plusieurs projets reposent d'ores et déjà dessus : une intégration dans Dreamweaver[24], un plug-in pour Brackets, etc. L'usage détaillé de cette API[25] sort du cadre de cet ouvrage. Sachez simplement qu'elle permet aujourd'hui d'agir en lecture et écriture sur la plupart des réglages et autres fonctionnalités présentées précédemment. On retrouvera notamment :

- l'authentification d'utilisateurs ;
- l'accès aux applications et leur création/suppression ;
- le déclenchement de la compilation pour des plates-formes données ;
- la gestion des collaborateurs, ainsi que des clés de sécurité et même leur déverrouillage.

Via le terminal

La dernière façon de profiter des vertus de PhoneGap Build est d'utiliser l'outil en ligne de commande PhoneGap. Ce dernier, tirant parti de l'API REST mentionnée précédemment, prend en charge le service ; bien qu'encore assez sommairement. Tâchons de décrire chacune des commandes disponibles. Ouvrez une nouvelle fenêtre de terminal dans le répertoire de votre choix. Puis, pour créer une nouvelle application basée sur le squelette par défaut située dans le sous-répertoire `my-app`, entrez :

```
phonegap create my-app && cd my-app
```

23. http://fr.wikipedia.org/wiki/Representational_State_Transfer
24/ http://www.adobe.com/fr/products/dreamweaver.html
25. http://docs.build.phonegap.com/en_US/developer_api_api.md.html#PhoneGap%20Build%20Developer%20API

Copyright © 2015 Eyrolles.

Exécutez ensuite :

```
phonegap help remote
```

Vous aurez ainsi accès à la liste des commandes à votre disposition, de leurs arguments et options.

Note

Seul le téléchargement d'archives au format ZIP est actuellement permis par l'interface en ligne de commande. Selon la nature de votre compte utilisateur, assurez-vous de posséder au moins un espace libre pour la création d'applications privées.

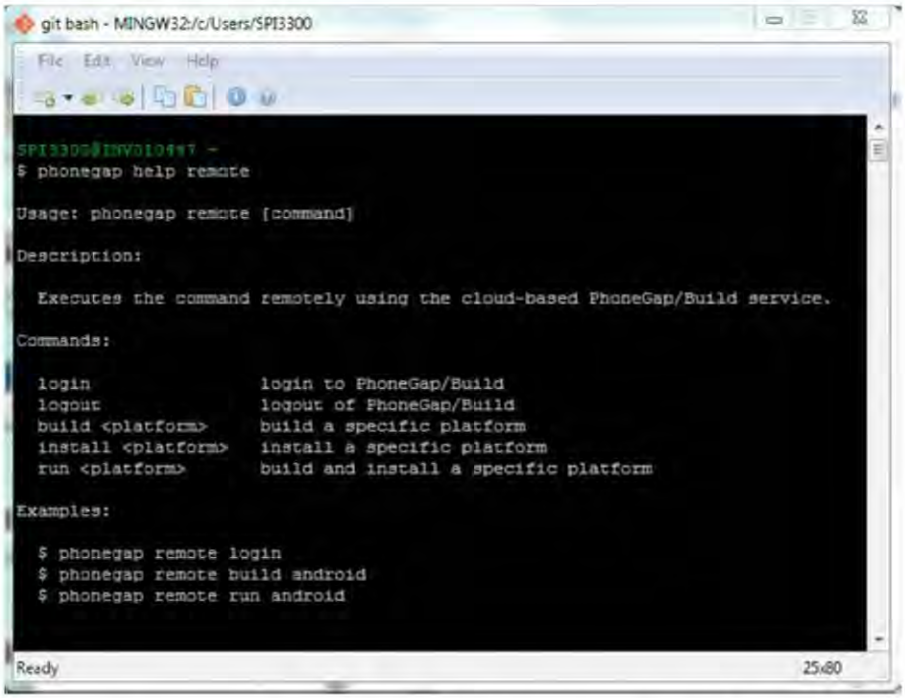


Figure 8-26
phonegap help remote

Après avoir effectué les changements souhaités dans le fichier `config.xml`, entrez la commande `phonegap remote login`, suivie de votre couple identifiant/mot de passe. Vous voilà maintenant connecté à PhoneGap Build depuis votre terminal ! Pour l'action inverse, utilisez `phonegap remote logout`.

Afin de compiler l'application en cours, exécutez la commande `phonegap remote build <platform>`, où la valeur de `platform` peut être `ios`, `android` ou `wp8`. Patientez pendant le téléchargement de votre application. Un message vous préviendra lorsque la comppilation sera effective, précisant si d'éventuelles erreurs ont eu lieu. La commande `phonegap remote install <platform>`, quant à elle, permet d'installer une application déjà compilée sur des appareils de test. Pour cela, elle utilisera soit les différents SDK présents sur votre machine, soit affichera le code QR associé directement dans votre terminal. Enfin, la commande `phonegap remote run <platform>` enchaîne simplement l'exécution des deux précédentes.

Copyright © 2015 Eyrolles.

C'est tout pour la version actuelle et c'est peu en regard des multiples possibilités offertes, aussi bien par l'interface web que l'API REST. On regrettera notamment l'absence de gestion des clés de sécurité. Gageons que l'outil, comme il l'a très bien fait depuis ses débuts, évoluera vite pour combler ces manques. Son code source est en accès public sur GitHub (https://github.com/phonegap/phonegap-cli), et est donc ouvert à tous ceux qui désireraient contribuer à son amélioration.

Support et distribution

S'il est possible d'être bloqué avec une compilation qui échoue constamment, l'équipe de PhoneGap Build a recensé[26] les erreurs les plus courantes. Pouvant découler d'un problème technique sur le service (http://status.build.phonegap.com/), ou encore de la malformation d'un fichier important, celles-ci peuvent être totalement dépendantes de l'implémentation du code pour une plate-forme donnée. On saluera l'initiative. Consulter cette liste est vivement conseillé avant de se lancer dans l'aventure. Tout comme la page présentant les différents moyens d'entrer en relation avec les développeurs d'Adobe et de la communauté[27].

Après avoir signé une application avec les clés de distribution nécessaires et récupéré les paquets compilés par PhoneGap Build, soumettre celle-ci sur les magasins d'applications en ligne est relativement aisé. En effet, il suffit de suivre les mêmes procédures que celles décrites dans le chapitre 7 (sections « Publication via iTunes Connect » et « Création et mise en ligne de l'application », pages 133 et 143). Après tout, il ne s'agit ni plus ni moins que d'une application Cordova/PhoneGap classique pour laquelle vous n'avez simplement pas eu à gérer le processus de compilation.

26 http://docs.build.phonegap.com/en_US/support_failed-builds.md.html#Failed%20Builds
27 http://docs.build.phonegap.com/en_US/support_getting_help.md.html#Getting%20Help

Copyright © 2015 Eyrolles.

Conclusion

Vous voici arrivé au terme de notre vue d'ensemble du développement mobile avec Cordova/PhoneGap.

Vous avez à présent acquis les bases pour développer votre propre application mobile qui, on vous le souhaite sincèrement, atteindra peut-être le haut des classements des magasins en ligne ! En effet, utiliser des technologies web peut, contrairement à certaines idées reçues, conduire au succès. Tout comme produire un code 100 % natif ne garantit pas l'accès aux sommets.

L'éternel débat natif contre hybride agite la communauté des développeurs mobiles depuis très longtemps et continuera, sans doute, pendant encore de nombreuses années. Aucune solution n'est forcément meilleure qu'une autre, il s'agit simplement de connaître les différents types d'utilisation afin de faire le bon choix.

Par exemple, on privilégiera généralement le développement Cordova/PhoneGap dans les cas où l'on souhaite être présent sur plusieurs plates-formes à moindres frais, rapidement, et là où d'excellentes performances ne sont pas requises. La principale difficulté étant ici d'offrir aux utilisateurs de votre application une interface/expérience agréable à l'œil, fluide, ergonomique et efficace. S'il existe quelques frameworks et autres bibliothèques pouvant vous aider dans cette tâche, la plupart sont destinés au Web et encore peu aux applications mobiles. Le plus gros du travail pour réaliser un produit de qualité consistera donc à affiner des détails néanmoins importants. C'est le principe de Pareto[1].

Dans cet ouvrage, nous avons étudié un peu de Cordova, un peu de PhoneGap, et un peu du développement mobile. Or, vous vous en doutez, il reste beaucoup à dire, énormément de choses à savoir et à faire dans cet univers passionnant. À vous d'en explorer maintenant les limites ! En plus des concepts et fonctionnalités que nous n'avons pas couverts, tous les jours se créent de nouveaux frameworks, de nouveaux langages, de nouvelles bonnes pratiques, etc. Le développement est un apprentissage constant aux nouvelles techniques : les outils et plates-formes cités au fil de cet ouvrage évoluent toujours très vite et il existe bien souvent plusieurs façons d'arriver à un même résultat. Seules une veille et une pratique assidues vous permettront de vous maintenir à niveau.

Comme nous l'avons précisé dans l'avant-propos, le code de l'application « Rappelle-toi » est disponible en ligne sur GitHub et en miroir sur www.editions-eyrolles.com. Nous pourrons ainsi facilement le faire évoluer pour l'améliorer ou corriger les bugs qui ne manqueront pas d'apparaître.

1. http://fr.wikipedia.org/wiki/Principe_de_Pareto

Copyright © 2015 Eyrolles.

Nous vous remercions d'avoir partagé cette aventure avec nous. N'hésitez pas à nous faire part de vos progrès, remarques, retours, critiques et, bien sûr, de vos succès.

Bon développement !

Sébastien Pittion
Compte Twitter : @fingerproof

Bastien Siebman
Compte Twitter : @siebmanb

Copyright © 2015 Eyrolles.

Index

A

ADT 30, 140
APK 140
app.js 74
App Store 6, 27, 129, 133

B

Bash 17, 20
bibliothèque
  JavaScript 51
    AngularJS 51
    Backbone 51
    Fastclick 51
    jQuery 51
    jQuery Mobile 51
    RequireJS 51
    Sencha Touch 51
  UI 51
    ChocolateChip-UI 51
    Flat UI 51
    Font Awesome 52
    Glyphicon 52
bonnes pratiques 77

C

Chrome Dev tools 107
  Audits 111
  Console 108
  Elements 107
  Emulation 109
  Network 109
  Profiles 110
  Resources 111
  Sources 109
CLI 15, 38, 39
ConfiGAP 162, 167
  icônes et splashscreens généraux 164
  paramètres
    avancés 163
    généraux 162
  permissions et informations 165
  Plugins (vue) 166
config.xml 53, 70, 150, 158, 175
  contenu minimal 151
  personnaliser 69
cordova serve 38, 95, 96, 106
CSS 50, 94, 108, 112

D

débogage
  Chrome Dev tools 107
  dans un navigateur 105
  dans un simulateur 113
    Android 117
    iOS 114
  émuler un appareil mobile 111
  sur un appareil 123
    iOS 123
    weinre 124
déploiement ad hoc 139

Copyright © 2015 Eyrolles.

F

fichiers CSS 71
  de style 71
  externes 71
  main.css 71
  theme.css 71

G

GenyMotion 114
géolocalisation (ajouter une) 75
Git 17, 58
  commandes de base 59
  fonctionnement 58
GitHub 47, 58, 60
  installer les plug-ins 79
  lié à PhoneGap Build 150
Google Play Store 6, 140

H

hooks 78, 81, 95, 97
HTML 5 7, 8
Hydration 175-177

I

icône 98, 99, 101, 162
index.html 71
IPA (fichier) 140, 156
iTunes Connect 115, 129, 133

L

local repositories 58

M

main.js 74
Member Center 129, 130, 138, 140
  Devices 132
  iOS App IDs 131
Mercurial 58
merges 81, 94, 106
  dossier par défaut d'un projet 53

N

Node.js 23-26
npm 25, 26

P

PhoneGap Build 12, 39, 149
  Debugging et Hydration 176
  fonctionnalités 158
  Hydration 150
  icônes et splashscreen 156
  paramètres généraux 151
    Android 153
    iOS 152
  personnalisation 154
  plug-ins 159
  schéma d'URL 158
  sécurité 161
  support et distribution 180
  utilisation 167
    via l'API REST 178
    via le terminal 178
    via une interface web 168
photo (ajouter une) 75
platforms (dossier par défaut d'un projet) 53
plug-in 10, 37-39, 97
  ajout 69
  Battery Status 46
  Camera 46
  Console 46
  Contacts 46
  Cordova
    Battery Status 90
    Camera 82
    Device 81
    Dialogs 86
    Geolocation 84
    InAppBrowser 91
    installer 79
    Network information 88
    plug-ins officiels 46
    SplashScreen 81
    Status Bar 90
  Device 46

Copyright © 2015 Eyrolles.

Device Motion 46
Device Orientation 46
Dialogs 46
File 46
File Transfer 46
fonctionnement 80
Geolocation 46
Globalization 46
identification 66
In-App Browser 46
installation manuelle 80
Media 47
Media Capture 47
Network Information 47
SplashScreen 47
Statusbar 47
sur GitHub 47
Vibration 47
plugins (dossier par défaut d'un projet) 53
plugman 37, 80
installer les plug-ins 80
profil ad hoc 133
projet Cordova
architecture type 53
création 69
fichier de configuration 53
provisionning profile 140

R

remote repository 58
repository Cordova
installer les plug-ins 79
repository (création) 66
repository local
installer les plug-ins 80
retina 112

S

Safari 107, 116, 117, 123, 124, 158
shell 16, 100
splashscreen 97, 101, 135, 144, 154, 156, 158, 164
Subversion 58
SVN 58

T

TestFlight 140

V

versioning 57
outils 58
Git 58
Mercurial 58
SVN 58
vidéo (ajouter une) 76

W

weinre 122-124, 159, 176
www 53, 56, 57, 73, 94

Copyright © 2015 Eyrolles.

N° d'éditeur : 9421
Dépôt légal : avril 2015
Imprimé en France chez Soregraph